MASTER EN SCIENCES INFORMATIQUES

Mise en place d'un référentiel Web des méthodes et pratiques agiles

Detaille, Thomas

Award date:
2015

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2014 – 2015

Mise en place d'un référentiel Web des méthodes et pratiques agiles

Thomas Detaille

Promoteur : _____ (Signature pour approbation du dépôt – REE art. 40)
Naji Habra

Co-promoteur : Hajer Ayed

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

L'adoption des méthodes agiles connaît un engouement certain. De plus en plus d'organisations voient l'agilité comme une solution aux problèmes auxquels elles sont confrontées. Dès lors, celles-ci décident de franchir le cap et procèdent à l'adoption partielle ou complète d'une ou plusieurs méthodes agiles.

Fort de ce constat, nous avons souhaité mettre en place un outil permettant de définir un cadre de référence afin de soutenir l'adoption massive de l'agilité qui s'opère.

Ce travail a pour objectif principal de communiquer la démarche entreprise pour concevoir et réaliser un prototype logiciel d'un référentiel web sur le thème des méthodes et pratiques agiles.

Un tel outil a pour but primaire de fournir une information claire, structurée et non partisane sur les diverses méthodes et pratiques agiles couramment utilisées.

mots clés: agilité, adoption, méthode agile, pratique agile, référentiel, Scrum, XP

Abstract

The adoption of agile methods and practices knows a sure infatuation. More and more organizations consider the agility like a solution for the problems they meet. That is the reason why they decide to take the step and they choose the partial or the complete adoption for one or more agile methods.

In consequence of this fact we wish to set up a framework in order to support the massive adoption of agility.

This work has for principle target to communicate the way to design and implement a software prototype of a web referential on the topic of agile methods and practices.

Such a tool has for primary objective to furnish clear, structured and non-partisan information on the varied methods and practices commonly used.

keywords: agility, adoption, agile method, agile practice, referential, Scrum, XP

Avant-propos

Je tiens particulièrement à remercier:

Monsieur Naji Habra, Doyen de la Faculté d’Informatique, pour avoir accepté d’être le promoteur de ce mémoire, ainsi que pour sa disponibilité et ses précieux conseils.

Madame Hajer Ayed pour son aide et ses suggestions pertinentes tout au long de ce travail.

Ma famille qui, par sa patience et son soutien moral, m'a permis d'aller jusqu'au terme de ces années d'études.

Mes amis, mes collègues et toutes les personnes qui de près ou de loin m'ont supporté lors de la réalisation de ce travail.

Je remercie également les membres du Jury de l'intérêt qu'ils voudront bien porter à la lecture de ce travail.

Table des matières

Table des figures

Table des tableaux

Glossaire

AJAX :

Acronyme pour « Asynchronous JavaScript And XML ». Technologie pour construire des pages web dynamiques côté client. Les données sont échangées avec le serveur via des requêtes JavaScript et le serveur effectue des traitements sur ces données.

Back-end :

Partie « privée » d'un site ou d'une application web.

Background :

Antécédents de quelqu'un, notamment en ce qui concerne ses études et sa carrière.

Bug :

Anomalie de fonctionnement dans une application.

Framework :

Ensemble de bibliothèques et de conventions permettant le développement rapide d'applications.

Front-end :

Partie « publique » d'un site ou d'une application web (interface entre l'utilisateur et le système).

Manifeste :

Texte qui est considéré comme étant à l'origine d'un nouveau mouvement.

Output :

Résultat d'une production.

Plugin :

Module d'extension qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Release :

Nouvelle version d'une application comportant des améliorations.

Time to market :

Délai représentant le temps que met un produit à être commercialisé sur le marché.

Introduction

Ce travail a été réalisé en vue de l'obtention du diplôme universitaire de master 60 à horaire décalé en sciences informatiques effectué au sein de l'université de Namur.

Objectif

L'objectif principal de ce travail consiste à communiquer la démarche entreprise afin de réaliser un prototype logiciel d'un référentiel web permettant de documenter les méthodes et pratiques agiles.

Source d'information

Les sources d'informations ayant contribué à la réalisation de ce travail sont diverses. Nous avons recouru à différents ouvrages principalement issus de la recherche scientifique, ainsi qu'à une multitude de documents et d'articles consciencieusement sélectionnés et disponibles sur internet. Notons que pour les informations relatives aux méthodes agiles, nous nous sommes principalement basés sur les publications des auteurs de ces dernières.

Méthodologie

Afin de répondre à notre objectif, nous avons suivi une démarche en cinq étapes :

- 1ère étape : Présentation du contexte et de la problématique ;
- 2ème étape : Analyse de quelques méthodes agiles ;
- 3ème étape : Réalisation d'un prototype logiciel d'un référentiel web sur le thème des méthodes et pratiques agiles ;
- 4ème étape : Présentation du prototype réalisé ;
- 5ème étape : Critique du prototype réalisé.

Structure du mémoire

La structure de ce document suit cette démarche.

Le premier chapitre est consacré à la présentation du contexte et de la problématique. Nous y présentons les concepts clés abordés dans ce travail, introduisons la notion d'agilité en citant le manifeste agile, définissons les méthodes agiles et leurs enjeux, abordons le thème de l'adoption des méthodes agiles et énonçons la problématique de ce travail.

Le second chapitre énumère quelques-unes des principales méthodes agiles existantes et présente leurs fondements.

Le troisième chapitre est dédié à la réalisation d'un prototype logiciel d'un référentiel web sur le thème des méthodes et pratiques agiles. Au sein de ce chapitre, nous introduisons d'une part la notion de référentiel en proposant une définition et en citant

les intérêts principaux et les qualités primordiales d'un tel outil. D'autre part, nous présentons les points saillants et les choix opérés quant à l'analyse, la spécification et la conception du prototype.

Le quatrième chapitre présente l'outil créé sur base des spécifications.

Le cinquième chapitre est consacré à la critique du prototype. Dans ce chapitre, nous mettons en avant les points forts de l'application, les améliorations et extensions possibles, ainsi que ses perspectives d'utilisation.

Finalement, nous concluons en passant en revue les différentes phases de notre travail.

1 Contexte et problématique

Ce chapitre a pour objectif de présenter le contexte et la problématique de ce travail. Nous commençons par définir les concepts clés rencontrés dans cet ouvrage. Ensuite, nous abordons la notion d'agilité. Pour cela, une présentation du manifeste agile est réalisée, les méthodes agiles sont définies et leurs enjeux sont énoncés. Après, nous continuons en abordant le thème de l'adoption de ces méthodes. Finalement, nous terminons par la formulation de la problématique.

1.1 Définition des concepts clés

La définition d'un terme peut varier en fonction du contexte et du domaine d'application. Dès lors, nous aimerions vous inviter à découvrir le sens des principaux concepts abordés dans ce travail afin d'éviter toute confusion lors de la lecture de ce dernier.

1.1.1 Approche

Une approche est une « manière d'aborder un sujet, un problème ». [Larousse, 2015]

Notons qu'une approche est couramment guidée par des valeurs et des principes. [Boisvert et al, 2011]

1.1.2 Méthodologie

La méthodologie est l'étude des « méthodes et des techniques d'un domaine particulier ». [Larousse, 2015 ; Coq, 2012]

1.1.3 Méthode

De manière générale, une méthode est une façon de procéder, un « ensemble de moyens raisonnés permettant de parvenir à un résultat ». [Académie Française, 2015]

1.1.4 Pratique

Une pratique est une « application, exécution, mise en action des règles, des principes d'une science, d'une technique ». [Larousse, 2015]

1.1.5 Artéfact

Un artéfact est un élément concret produit ou modifié dans le cadre d'un processus. [Epf.eclipse.org, 2008]

1.2 Le manifeste agile

1.2.1 Qu'est-ce que le manifeste agile ?

Le manifeste agile est un texte qui définit de manière claire et concise l'essence de l'approche agile au travers de quatre valeurs et douze principes fondateurs. Ce dernier est considéré comme la définition canonique du développement agile et de ses principes sous-jacents [Vickoff, 2009].

Notons que les méthodes agiles ne sont pas nées suite à la rédaction du manifeste agile. En effet, la plupart de ces méthodes ont fait leur apparition dans le courant des années 90 [Abrahamsson et al, 2002], c'est-à-dire, antérieurement au manifeste qui fut quant à lui rédigé en 2001 [Beck et al, 2001]. Le manifeste n'a donc pas donné naissance aux méthodes agiles mais formalisé l'approche en définissant le dénominateur commun de ces méthodes.

1.2.2 Origines du manifeste agile

Le manifeste agile a vu le jour aux Etats-Unis, en février 2001, dans l'état de l'Utah. Celui-ci a été rédigé, suite au consensus obtenu entre dix-sept spécialistes en génie logiciel, lors de la mise en commun de leurs idées, à propos de nouvelles méthodes de développement logiciel plus légères que celles alors en vigueur [Beck et al, 2001]. Parmi eux, figurent entre autres, Ken Schwaber et Jeff Sutherland, co-auteurs de la méthode Scrum.

Le texte résultant de cette déclaration expose les fondements de base de la philosophie agile en présentant les valeurs et principes généraux et communs à ces méthodes. Aujourd'hui, ces valeurs et principes sont soutenus, défendus et promus par l'organisation à but non-lucratif « Agile Alliance », fondée suite à cette convention et par cette même cohorte [Agile Alliance, 2015].

1.2.3 Les valeurs du manifeste agile

Le manifeste agile met en avant quatre valeurs fondamentales qui caractérisent l'approche agile. Celles-ci sont listées dans l'extrait suivant du manifeste :

> Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire. Ces expériences nous ont amenés à valoriser :
>
> **Les individus et leurs interactions** plus que les processus et les outils
> **Des logiciels opérationnels** plus qu'une documentation exhaustive
> **La collaboration avec les clients** plus que la négociation contractuelle
> **L'adaptation au changement** plus que le suivi d'un plan
>
> Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.

Figure 1 : Le manifeste agile [Beck et al, 2001]

La dernière phrase posée dans cet extrait détient toute son importance. Celle-ci met en évidence le fait que l'on valorise davantage les premiers éléments aux seconds. Néanmoins, cela ne signifie pas que l'on se passe intégralement de ceux-ci dans la philosophie agile.

1.2.4 Les principes du manifeste agile

Le manifeste agile définit douze principes sous-jacents aux valeurs citées précédemment. Ci-dessous est présentée la liste exhaustive de ces principes.

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
7. Un logiciel opérationnel est la principale mesure d'avancement.
8. Les processus agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité, c'est-à-dire l'art de minimiser la quantité de travail inutile, est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. A intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

Figure 2 : Principes sous-jacents au manifeste agile [Beck et al, 2001]

1.3 Les méthodes agiles

1.3.1 Qu'est-ce qu'une méthode agile ?

Les méthodes agiles sont des groupes de pratiques de développement logiciel et de gestion de projet qui suivent la philosophie agile, c'est-à-dire qui respectent les valeurs et principes listés au sein du manifeste agile.

Celles-ci prennent le contre-pied et offrent une alternative aux méthodes traditionnelles de type « V » ou « cascade » prônant un enchainement séquentiel des activités et le suivi d'un plan fixe préalablement défini.

L'Alliance Agile ne propose pas de définition formelle pour les méthodes agiles. Cependant, il existe de nombreuses tentatives de définition d'auteurs divers. Nous avons retenu celle proposée par Véronique Messager Rota dans l'ouvrage « Gestion de projet : Vers les méthodes agiles » [Messager Rota, 2009] :

« Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif avec juste ce qu'il faut de formalisme. Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients »

L'objectif de ce type de méthode est de faciliter la mise en œuvre du travail au sein d'un projet et de maximiser la satisfaction du client en générant rapidement, régulièrement et de manière flexible des livrables de haute qualité, grâce à un esprit orienté résultat, ainsi qu'une collaboration et une communication optimales entre les différents acteurs.

Notons qu'il existe de nombreuses méthodes agiles. Chacune comporte ses spécificités et apporte son propre lot de pratiques, les unes concernant plutôt le pilotage de projet, les autres, plutôt l'ingénierie logicielle. Toutefois, toutes respectent le paradigme définit par le manifeste agile.

1.3.2 Une approche itérative et incrémentale

Les méthodes agiles proposent un processus itératif et incrémental afin de pouvoir s'adapter facilement aux changements en cours de projet et d'offrir au client une visibilité accrue et un contrôle plus flexible sur le produit et son évolution.

Précisons que, par itératif et incrémental, on entend que le produit est réalisé morceau par morceau, sous forme de cycles successifs jusqu'à l'obtention du résultat attendu. Chaque cycle permet d'ajouter de nouveaux éléments, de corriger, mais aussi d'améliorer des éléments existants du produit. Chaque cycle a pour objectif de produire une version intermédiaire, stable et fonctionnelle de ce dernier.

La figure suivante (Figure 3) schématise de manière visuelle l'approche itérative et incrémentale. Dans cette figure, on note trois itérations. Durant chacune d'entre elles, un

incrément utilisable et potentiellement livrable est produit. Notons que si le projet devait être arrêté de manière prématurée, et ce dès la fin de la première itération, nous ne disposerions certes pas d'un buste de la Joconde, mais nous pourrions néanmoins jouir d'un portrait de celle-ci [Vickoff, 2014].

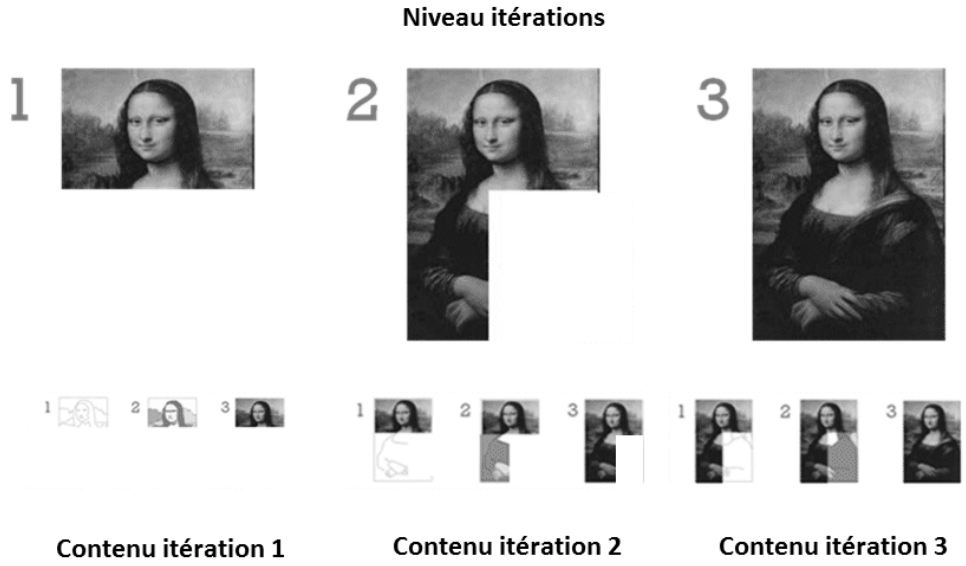


Figure 3 : L'itératif combiné à l'incrémental [Vickoff, 2014]

1.4 Les enjeux des méthodes agiles

A l'ère de la mondialisation et de la digitalisation, où l'adaptabilité à un monde changeant est maître mot, où la gestion des projets devient de plus en plus complexe et où le time to market tend à se réduire, les méthodes traditionnelles de type « V » ou « cascade » ne permettent plus toujours de parvenir aux résultats escomptés.

L'accélération des innovations technologiques, une concurrence internationale accrue ou encore la rapidité d'évolution du contexte économique, social et politique, sont divers facteurs pouvant mener à des changements mineurs ou radicaux dans les plans ou encore la stratégie d'une entreprise. Tout cela peut bouleverser les besoins des projets en cours ou à venir, ainsi que leur priorité.

Ces divers facteurs organisationnels et environnementaux engendrent une nécessité de changement et d'évolution continue des besoins, tant bien fonctionnels que techniques, à laquelle les méthodes traditionnelles ne peuvent répondre efficacement.

Etre le premier face à la concurrence ou récolter rapidement un premier retour sur investissement sont souvent des facteurs critiques pour une entreprise dans le contexte économique actuel. Ces objectifs sont parfois freinés ou mis à mal suite à la rigidité et à la durée du cycle de vie trop long de la méthodologie adoptée.

La méthodologie agile prend le contre-pied aux méthodes traditionnelles et tente de faire face à ces problématiques en proposant un cadre de travail plus souple axé sur la collaboration, la communication, l'inspection, une amélioration constante des pratiques et processus, un cycle de vie réduit permettant de minimiser les risques et de raccourcir les délais de mise sur le marché des produits. Tout ceci avec, pour but ultime, de mieux appréhender le changement et de maximiser le retour sur investissement et la satisfaction du client.

1.5 L'adoption des méthodes agiles

L'adoption des méthodes agiles est actuellement en pleine effervescence. On constate que de plus en plus d'organisations entreprennent une transition vers l'agilité. Parmi celles-ci, figurent des géants du web comme Amazon, Google et Facebook, ou encore des institutions financières telles que BNP Paribas, ING et KBC.

La neuvième édition de l'étude annuelle sur l'état de l'agilité « State of Agile » confirme ce constat et indique que seulement 5 % des 3925 répondants travaillent au sein d'organisations pour lesquelles aucune équipe n'exerce ses activités de développement logiciel de manière agile contre 31 % en 2009. Cette même étude indique notamment que 45% des répondants travaillent dans un milieu où la majorité de leurs équipes opèrent de manière agile [VersionOne, 2014].

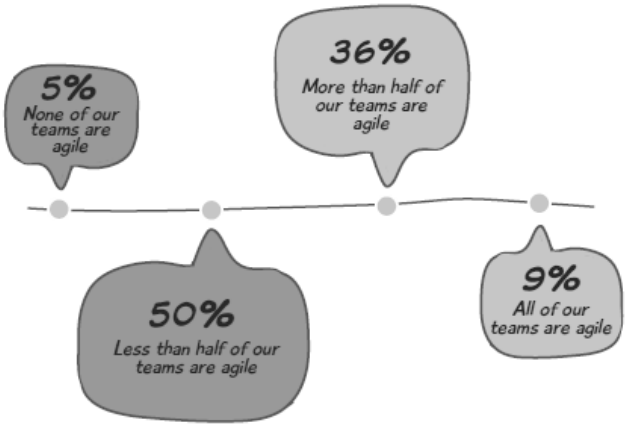


Figure 4 : Statistiques - Degré d'agilité en entreprise en 2014 [VersionOne, 2014]

1.5.1 Les principales méthodes agiles adoptées

Parmi les méthodes agiles les plus utilisées, « Scrum » se trouve largement en tête, suivie par la combinaison de cette dernière avec la méthode « eXtreme Programming ». A l'inverse, les méthodes « Dynamic Systems Development Method », « Agile Unified Process », ou encore « Feature-Driven Development » ont plus de mal à s'imposer. [VersionOne, 2014]

Un autre fait qui mérite d'être mis en avant, est que pas mal d'entreprises utilisent des méthodes customisées et/ou hybrides. On entend par « customisée », une méthode sur mesure adaptée au contexte organisationnel de l'entreprise. Celle-ci peut être simplement une combinaison de plusieurs méthodes agiles existantes, l'adaptation d'une méthode agile existante, ou encore la formation d'une nouvelle méthode à partir de pratiques agiles existantes.

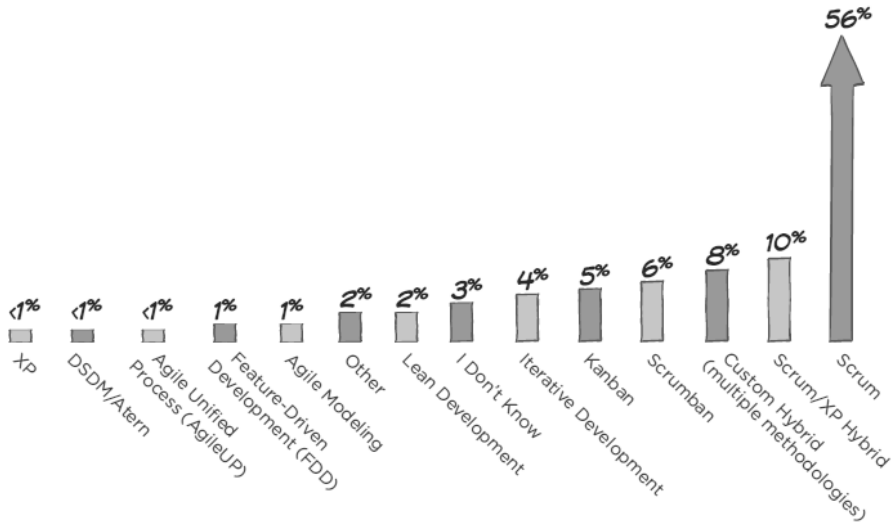


Figure 5 : Statistiques - Les principales méthodes agiles utilisées [VersionOne , 2014]

1.5.2 La combinaison de méthodes

La combinaison de méthodes est un concept important de l'adoption des méthodes agiles. L'idée est de mixer plusieurs méthodes agiles ou non afin d'obtenir une méthode hybride adaptée au contexte organisationnel de l'entreprise [Boisvert et al, 2011 ; West et al, 2010].

Un exemple concret que l'on rencontre fréquemment dans le domaine du développement logiciel est la combinaison des méthodes Scrum et XP [Boisvert et al, 2011 ; VersionOne, 2014]. Cette dernière offre un cocktail intéressant de pratiques du point de vue de la gestion de projet et de l'ingénierie logicielle.

Lors de la combinaison de plusieurs méthodes, il n'est pas impératif de retenir et d'appliquer l'intégralité des pratiques de ces dernières. En effet, il est tout à fait possible de choisir et combiner un sous-ensemble des pratiques de ces méthodes [Boisvert et al, 2011].

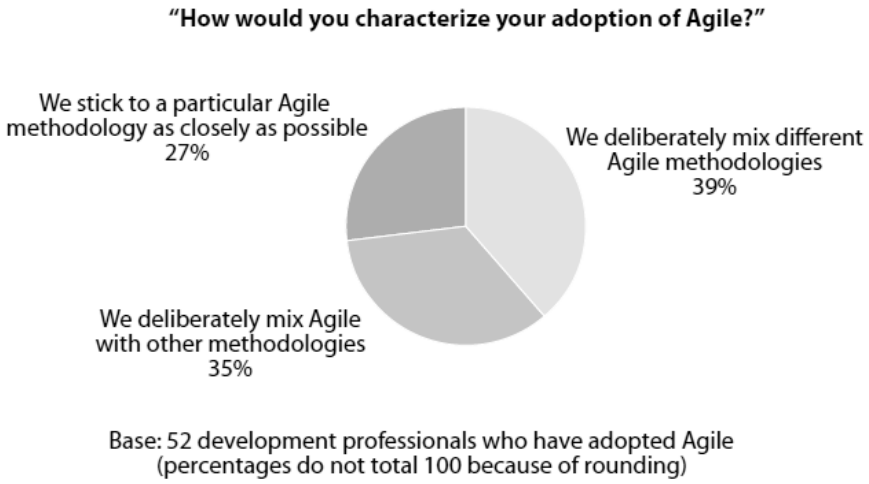


Figure 6 : Statistiques - La combinaison de méthodes [West et al, 2010]

1.5.3 L'adoption partielle des pratiques

L'adoption partielle des pratiques consiste à procéder à l'adoption des pratiques d'une méthode de manière progressive. Il est important de noter que pour être efficace, le choix et l'ordonnancement des pratiques à adopter ne doit pas être réalisé de manière aléatoire. En effet, la sélection doit être effectuée selon une stratégie guidée par des objectifs d'améliorations préalablement définis [Boisvert et al, 2011 ; Elssamadisy, 2009].

Amr Elssamadisy traite le sujet de l'adoption partielle des pratiques dans son ouvrage « Agile Adoption Patterns : A road to organizationnal success » où il propose des schémas de stratégies d'adoptions partielles basées sur des objectifs d'améliorations. Par exemple, la Figure 7 illustre les pratiques que l'auteur propose de sélectionner et l'ordonnancement d'adoption qu'il propose de suivre lorsque l'objectif est la réduction du time to market.

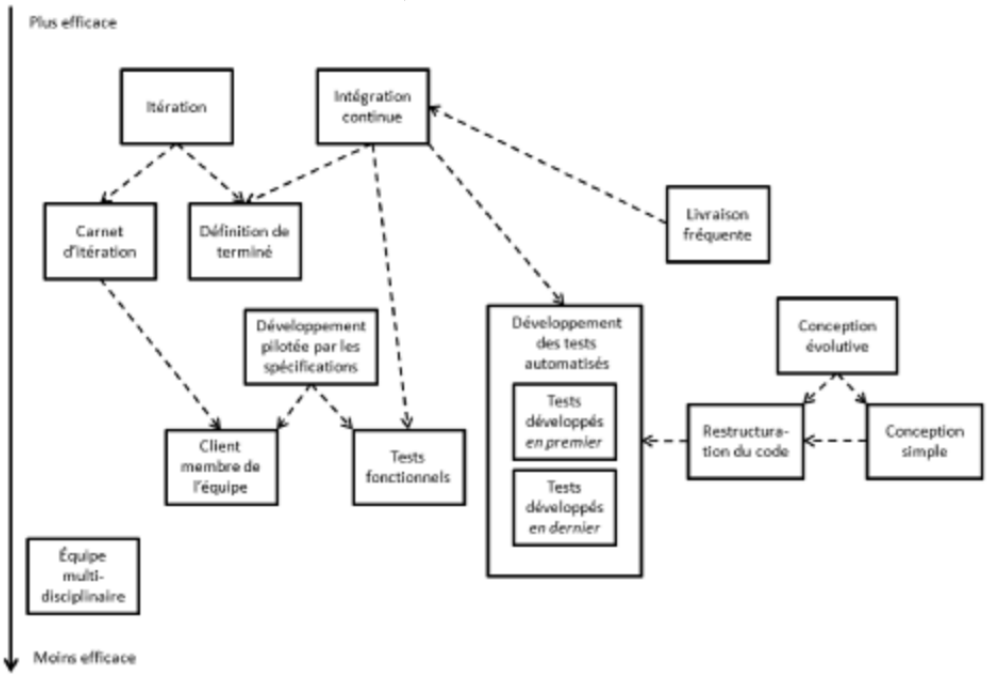


Figure 7 : Schéma d'adoption partielle pour réduire le time to market [Elssamadisy, 2009]

1.5.4 Les attentes constatées

Les attentes les plus communes quant à l'adoption des méthodes agiles sont dans l'ordre [VersionOne, 2014] :

1. « La réduction du délai de mise sur le marché des produits (time to market) »
2. « L'amélioration de la capacité de gestion du changement »
3. « L'augmentation de la productivité des équipes »

1.5.5 Les bénéfices constatés

Les principaux bénéfices constatés suite à l'adoption des méthodes agiles sont dans l'ordre [VersionOne, 2014] :

1. « L'amélioration de la capacité de gestion du changement »
2. « L'augmentation de la productivité des équipes »
3. « L'amélioration de la visibilité des projets »

Notons que ce top 3 reste inchangé depuis quatre années consécutives, et que ceux-ci sont en adéquation aux attentes des entreprises quant à l'adoption des méthodes agiles.

1.6 Problématique

Depuis plusieurs mois, nous contribuons à un projet de transition vers l'agilité pour une large institution financière dans le cadre de nos activités professionnelles[1]. Jusqu'ici, plusieurs projets pilotes ont été entrepris au sein du département informatique. Les résultats étant positifs, nous avons amorcé la seconde phase du processus d'adoption qui consiste à étendre la méthodologie à l'ensemble des équipes et des projets du département. Dès lors, les premiers problèmes quant à la bonne compréhension de la méthode et des pratiques sont apparus. Ce type de problème n'est pas spécifique à notre projet, mais d'ordre général.

C'est pourquoi, dans le cadre de ce travail, nous souhaitons répondre à cette problématique et proposer un élément de solution pour y faire face. Pour cela, nous avons pensé à la mise en place d'un référentiel web destiné à la communauté agile. Le but premier de ce référentiel est de permettre à des experts en méthodologie de centraliser et publier des informations utiles et véraces à propos des diverses méthodes et pratiques agiles existantes afin que tout un chacun puisse s'y référer.

Notons, qu'en enrichissant le référentiel avec diverses informations telles que le domaine d'application, ou encore les bénéfices de l'emploi d'une méthode ou d'une pratique, celui-ci pourra également servir d'outil d'aide à la décision dans le cadre du choix de la méthode et/ou des pratiques agiles à adopter. Ce dernier aura un intérêt particulier dans le cadre de l'adoption partielle des pratiques agiles, où la sélection, l'agencement et la priorisation des pratiques est crucial.

[1] Conseil en informatique – Transformation digitale & Amélioration de la performance

2 Description de quelques méthodes agiles

Ce chapitre offre un tour d'horizon sur quelques-unes des principales méthodes agiles existantes. A cette fin, trois méthodes agiles ont été retenues: la méthode Scrum, la méthode eXtreme Programming et la méthode Open Unified Process.

Le paradigme agile regroupe plusieurs méthodes. Celles-ci reposent sur un ensemble de valeurs et principes communs. Néanmoins, chacune possède des singularités. En effet, la terminologie, le spectre d'application, ou encore le plan de mise en œuvre peuvent différer d'une méthode à une autre.

Au sein de ce chapitre, nous souhaitons mettre en avant trois de ces méthodes. De ce fait, nous présentons successivement « Scrum », « XP » et « OpenUP ». Le choix des deux premières méthodes est motivé par leur niveau de popularité élevé, tandis que la sélection de la dernière est, quant à elle, justifiée par un souci d'originalité.

2.1 La méthode « Scrum »

2.1.1 Présentation

La méthode Scrum est une approche agile, davantage axée sur la gestion de projet que sur le développement logiciel, qui offre un cadre de développement et de maintenance pour des produits complexes. Elle met l'accent sur des équipes auto-organisées et pluridisciplinaires qui créent un produit via plusieurs incréments, permettant ainsi d'améliorer continuellement le produit. Cette dernière a pour objectif de maximiser l'efficacité des équipes de gestion de projets et de faciliter la réponse aux changements durant la phase de réalisation. Scrum est, à l'heure actuelle, l'une des méthodes agiles les plus populaires.

Ken Schwaber et Jeff Sutherland définissent Scrum de la manière suivante [Schwaber et al, 2013] :

« Scrum : A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. »

2.1.2 Auteur(s)

La version initiale de Scrum a été créée dans le début des années 90 par Jeff Sutherland et Ken Schwaber. Ceux-ci formalisèrent la méthode en 1995 afin de la présenter à la conférence OOPSLA (« Object-Oriented Programming, Systems, Languages & Applications »).

2.1.3 Rôles et responsabilités

Dans la méthode Scrum, on identifie trois rôles distincts [Schwaber, 2004] :

- Product Owner ;
- Equipe de développement ;
- Scrum Master.

L'ensemble de ces rôles constitue l'équipe Scrum. Nous présentons, ci-dessous, ces rôles et les principales responsabilités qui leur sont liées.

Tableau 1 : Les rôles et responsabilités de la méthode Scrum [Schwaber et al, 2013]

Rôle	Responsabilités
Product Owner	• Création, suivi, maintien et hiérarchisation du Product Backlog • Assurance d'une bonne visibilité et compréhension du Product Backlog • Maximisation de la valeur du produit
Equipe de développement	• Auto-organisation et gestion de l'équipe et du travail à accomplir • Création, suivi et maintien du Sprint Backlog • Réalisation du travail nécessaire au sein d'un sprint à la production d'un incrément potentiellement livrable (accomplissement de l'objectif du sprint)
Scrum Master	• Garantie de l'apprentissage et du respect de la méthode • Assurance que les pratiques, valeurs et règles de la méthode sont comprises et mises en application • Identification, suivi de l'évolution et aide à l'élimination des obstacles • Protection de l'équipe vis-à-vis des perturbations extérieures

2.1.4 Pratiques

La méthode Scrum définit un certain nombre de pratiques. Ces dernières sont orientées sur la gestion du projet plutôt que sur le développement logiciel. Voici une description des principales pratiques composant la méthode.

Tableau 2 : Les pratiques de la méthode Scrum [Schwaber et al, 2013]

Pratique	Description
Sprint	Un sprint est un laps de temps imparti durant lequel un incrément de produit potentiellement livrable est réalisé. Il est recommandé que ce laps de temps n'excède pas quatre semaines afin de limiter la complexité et les risques. La durée d'un sprint reste constante durant toute la durée du projet. Généralement, la durée définie varie entre deux et quatre semaines. Un ensemble d'activités sont entreprises durant chaque sprint : • Une réunion de planification ; • Des mêlées quotidiennes ; • Des périodes de développement ; • Une revue de sprint ; • Une rétrospective de sprint. Un sprint correspond à une et une seule itération dans la mise en œuvre du produit. Un nouveau sprint est initié dès que le sprint précédent se termine.
Planification de Sprint **(Sprint Planning)**	La réunion de planification a pour objectif de planifier le sprint à entreprendre. C'est-à-dire, de définir le contenu à partir d'un sous-ensemble de fonctionnalités sélectionnées parmi la liste complète du Product Backlog, mais également de définir la façon de procéder à sa réalisation. Ce plan est mis en place par l'équipe Scrum au complet en abordant les deux questions suivantes : • « Que contiendra l'incrément livré lors du prochain sprint ? » • « Comment le travail nécessaire à la réalisation de l'incrément sera-t-il accompli ? » En terme de durée, elle varie de façon proportionnelle à la durée du sprint. Huit heures est le temps recommandé pour un sprint d'une durée de quatre semaines. Au terme de cette réunion, un Sprint Backlog définissant le contenu et le plan d'action de ce dernier est mis en place. L'équipe de

	développement est à même de présenter et d'expliquer celui-ci de façon claire et précise aux autres membres de l'équipe Scrum.
Mêlée quotidienne **(Daily Scrum)**	La mêlée quotidienne, comme son nom l'indique, est une réunion quotidienne. Celle-ci est limitée à quinze minutes tout au plus. Il est recommandé qu'elle tienne place à la même heure et au même endroit tout au long du projet. Le but de cette réunion est de faire le point sur l'avancement du travail et les difficultés rencontrées pour tout un chacun. Pour cela, chaque membre de l'équipe de développement est prié de répondre, de façon claire et concise, aux trois questions suivantes : • « Qu'ai-je accompli depuis la dernière session ? » • « Que prévois-je d'accomplir d'ici la session suivante ? » • « Quelles sont les difficultés rencontrées ? » Durant cette réunion, seule l'équipe de développement est invitée à participer. Tout autre intervenant éventuel est dépourvu du droit de parole. Celle-ci est menée par l'équipe de développement et encadrée par le Scrum Master.
Revue de sprint **(Sprint Review)**	La revue de sprint est une réunion de type informel prenant place entre tous les intervenants à la fin de chaque sprint. L'objectif est d'effectuer un bilan du travail accompli durant le sprint. Pendant cette réunion, l'incrément réalisé est présenté, discuté et critiqué. L'équipe de développement présente les réussites, les problèmes et les imprévus rencontrés. Le Product Owner, est quant à lui, chargé de déterminer ce qui est effectivement « terminé » ou non. En fonction des résultats, le Product Backlog peut, éventuellement, être revu et adapté. En terme de durée, celle-ci varie de façon proportionnelle à la durée du sprint. Quatre heures est le temps recommandé pour un sprint d'une durée de quatre semaines. Au terme de la revue de sprint, le Product Backlog est révisé de façon partielle ou complète. L'output de ce bilan doit apporter une valeur ajoutée à la planification du sprint suivant.
Rétrospective de sprint **(Sprint Retrospective)**	La rétrospective de sprint prend place après la revue du sprint. Cette réunion est tenue par l'équipe Scrum et a pour objectif de faire le point sur les forces et faiblesses de l'équipe, sur les problèmes rencontrés durant le sprint mais également sur le bon fonctionnement ou non des processus. Cette rétrospective est réalisée dans un but d'efficience et d'amélioration de la qualité des livrables sur base de l'expérience acquise au cours des itérations précédentes.

	En terme de durée, celle-ci varie de façon proportionnelle à la durée du sprint. Trois heures est le temps recommandé pour un sprint d'une durée de quatre semaines. En output de cette réunion, est attendu un plan d'amélioration qui sera mis en place durant le sprint suivant.

2.1.5 Artéfacts

Les auteurs de la méthode définissent, dans la version initiale, trois artéfacts majeurs [Schwaber, 2004] :

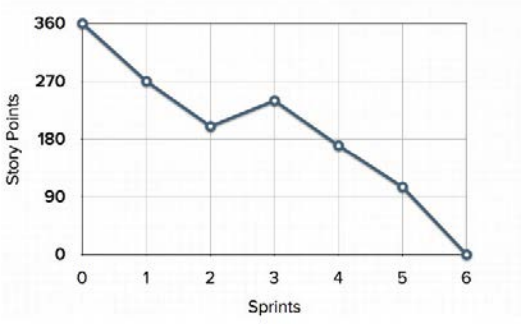
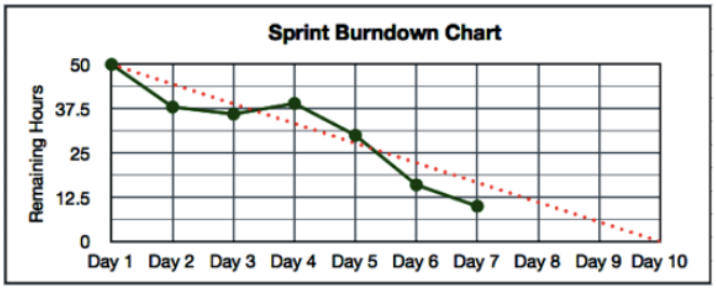
- Product Backlog ;
- Sprint Backlog ;
- Incrément.

Néanmoins, Ken Schwaber fait plus tard référence à deux artéfacts supplémentaires dans certaines de ses publications [Schwaber, 2009] :

- Release Burndown Chart ;
- Sprint Burndown Chart.

Tableau 3 : Les artéfacts de la méthode Scrum [Schwaber, 2004;Schwaber, 2009;Schwaber et al, 2013]

Artéfact	Description
Product Backlog	Le Product Backlog est une liste des fonctionnalités, besoins, améliorations et correctifs du produit. Cette liste est en constante évolution et généralement ordonnée selon la priorité, la valeur et le risque des éléments. Pour chaque élément, on associe une description, une estimation, ainsi qu'un ordre de priorité. Le niveau de détail de la description des éléments n'est pas uniforme. En effet, tous les éléments du Product Backlog ne possèdent pas forcément le même niveau de détails à un moment 'T'. Cependant, plus la phase de réalisation d'un élément est imminente, plus le niveau de détails de l'élément est important.
Sprint Backlog	Le Sprint Backlog définit l'objectif du sprint et son plan de réalisation. En d'autres mots, il spécifie les éléments du Product Backlog qui seront réalisés au sein du sprint et explicite les détails du travail nécessaire à leur élaboration.

Incrément	Un incrément est le résultat d'un sprint. Celui-ci correspond à un sous-ensemble d'éléments du Product Backlog, réalisé durant le sprint courant et ajouté à la réalisation des sprints précédents. Un incrément doit être utilisable, potentiellement livrable et doit répondre à la définition de « Terminé » convenue par l'équipe Scrum.
Release Burndown Chart	Le Release Burndown Chart est un outil graphique, mis à jour à la fin de chaque sprint, permettant de visualiser l'avancement global du projet. Celui-ci permet d'observer la charge de travail restante au sein d'une release au fil des sprints. Généralement, le travail restant est indiqué sur l'axe vertical et la période de temps sur l'axe horizontal.  **Figure 8 : Exemple de Release Burndown Chart**
Sprint Burndown Chart	Le Sprint Burndown Chart est un outil graphique, mis à jour quotidiennement, permettant de visualiser les progrès de l'équipe de développement. Celui-ci permet d'observer la charge de travail restante au sein d'un sprint, à un moment 'T', comparé à la charge et au planning initialement définis. Généralement, le travail restant est indiqué sur l'axe vertical et la période de temps sur l'axe horizontal.  **Figure 9 : Exemple de Sprint Burndown Chart**

2.2 La méthode « eXtreme Programming »

2.2.1 Présentation

La méthode eXtreme Programming, également connue sous l'acronyme « XP » est une méthode agile axée sur le développement logiciel. Celle-ci a pour objectif de maximiser la satisfaction du client en optimisant la réactivité au changement et la qualité des livrables. Afin de remplir son objectif, XP pousse à l'extrême des pratiques et principes simples du développement informatique. Cette méthodologie s'appuie sur cinq valeurs fondamentales : la communication, le courage, le retour d'information, la simplicité et le respect [Beck et al, 2005].

Ron Jeffries décrit XP de la manière suivante [Jeffries, 2011] :

« *Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.* »

2.2.2 Auteur(s)

La méthode XP a été inventée dans les années 90 par Kent Beck, en collaboration avec Ward Cunningham et Ron Jeffries lors de leur participation à un projet de gestion de paie pour la compagnie Chrysler. En 1999, la méthode fut formalisée par Kent Beck au sein de l'ouvrage « Extreme Programming Explained: Embrace Change » [Beck, 2000].

2.2.3 Rôles et responsabilités

La méthode XP définit plusieurs rôles. Nous présentons ici ces rôles et les principales responsabilités qui leur sont liées.

Tableau 4 : Les rôles et responsabilités de la méthode XP [Beck, 2000 ; Abrahamsson, 2002]

Rôle	Responsabilités
Programmeur	• Estimation des user stories ; • Définition et estimation des tâches nécessaires à la réalisation des user stories ; • Rédaction et maintenance du code ; • Rédaction et exécution des tests unitaires.
Client	• Spécification et priorisation des user stories ; • Spécification des tests fonctionnels.
Testeur	• Aide à la spécification des tests fonctionnels ; • Exécution des tests fonctionnels et diffusion des rapports de résultats.
Tracker	• Suivi de l'avancement des tâches ; • Détection des difficultés, problèmes et/ou déviations de trajectoire ; • Garantie de la précision des estimations.
Coach	• Garantie du respect de l'application et de la compréhension de la méthode.
Consultant	• Aide à la résolution de problèmes spécifiques.
Big Boss	• Obtention des ressources ; • Pilotage de l'équipe ; • Gestion des problèmes ; • Prise de décisions.

2.2.4 Pratiques

Kent Beck, auteur principal de la méthode, définit treize pratiques dites, principales ou primaires, et onze pratiques corollaires [Beck et al, 2005]. Nous nous focaliserons ici sur les pratiques primaires.

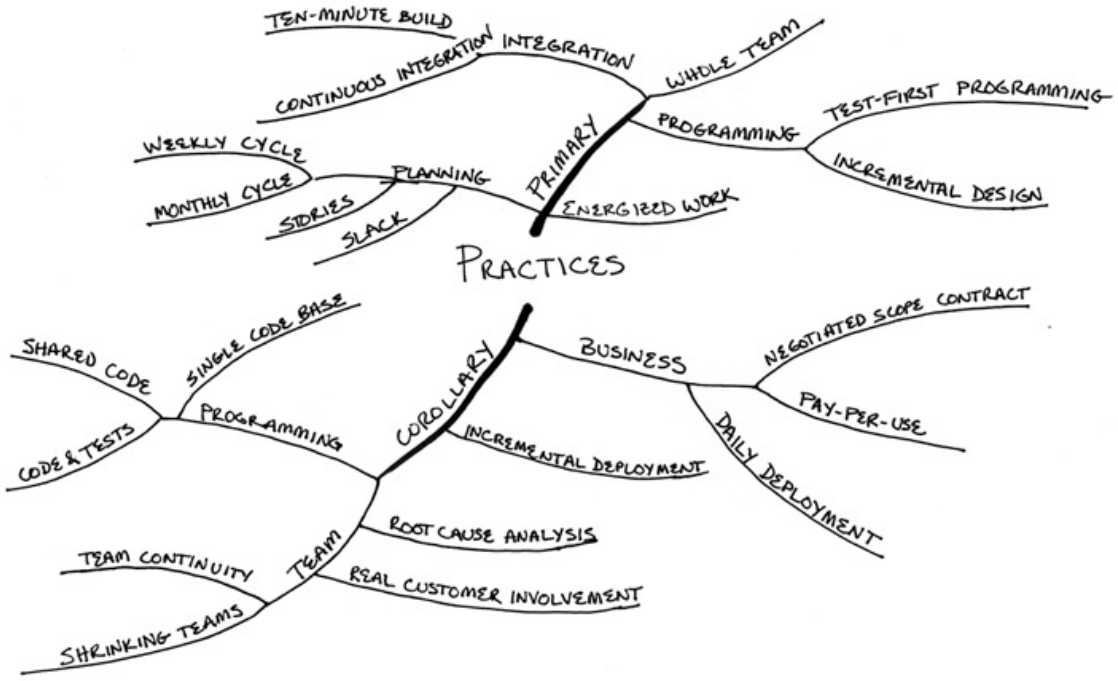
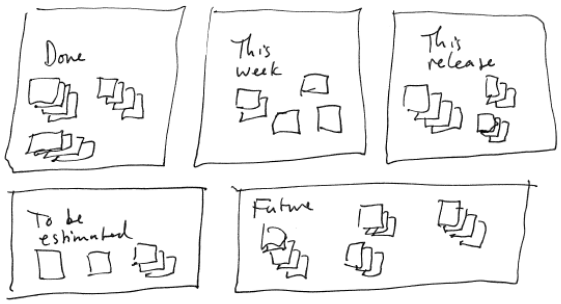
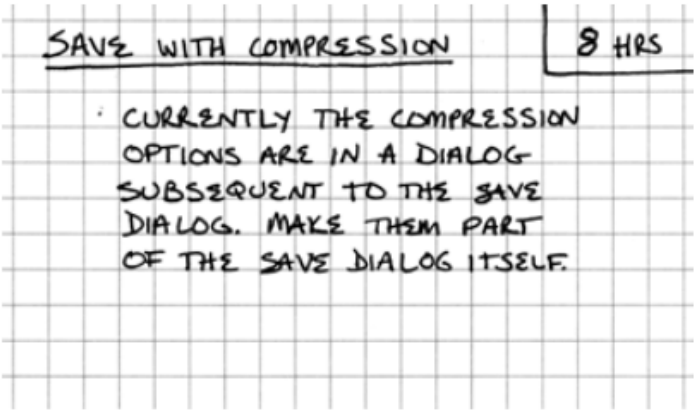


Figure 10 : Aperçu des pratiques de la méthode XP [Beck et al, 2005]

Tableau 5 : Les pratiques primaires de la méthode XP [Beck et al, 2005 ; Agile Alliance, 2015]

Pratique	Description
Espace commun de travail **(Sit Together)**	L'ensemble de l'équipe évolue dans un espace physique ouvert et commun suffisamment grand. L'objectif est de facilité la communication entre les individus d'un projet.
Equipe complète **(Whole Team)**	L'équipe est pluridisciplinaire. Celle-ci réunit l'ensemble des compétences nécessaires au bon déroulement du projet.
Management visuel **(Informative Workspace)**	L'espace de travail de l'équipe est arrangé de façon à ce que l'on puisse se faire une idée générale du déroulement du projet, en l'espace de quelques secondes. Les pratiques courantes pour ceci sont les suivantes : • Affichage et ordonnancement des user stories sur un mur ;

	• Présentation visible de graphiques concernant les progrès de l'équipe.  **Figure 11 : Exemple de user stories sur un mur [Beck et al, 2005]**
Travail dynamique **(Energized Work)**	L'équipe est astreinte à un horaire raisonnable permettant de tenir un rythme soutenu et une productivité optimale pendant toute la durée du projet.
Programmation en binôme **(Pair Programming)**	La programmation est réalisée par couples de développeurs. Chaque couple se partage un poste de travail unique. Le premier, dit pilote, écrit le code tandis que le second, nommé copilote, aide le premier, observe et détecte les erreurs éventuelles ou encore les améliorations possibles. Les binômes, ainsi que les rôles au sein de ceux-ci ne sont pas fixes. Des rotations régulières sont effectuées.
Stories	La planification des tâches est réalisée en utilisant des unités de fonctionnalité visibles par le client appelés user stories. Une user story décrit un besoin de manière brève et sommaire. Une user story est généralement écrite sous la forme d'un Post-It ou d'une fiche cartonnée, nommée story card, qui sera ensuite apposée sur un mur. Celle-ci est couramment composée des éléments suivants: • Un titre ; • Une description concise ou une simple représentation graphique ; • Une estimation.

	 **Figure 12 : Exemple d'une story card [Beck et al, 2005]**
Cycle hebdomadaire **(Weekly Cycle)**	Le travail à réaliser est planifié sur base hebdomadaire. Une réunion a lieu au début de chaque semaine. Le but de celle-ci est triple : • Statuer sur l'avancement et les progrès du projet ; • Définir le scope de la semaine en terme de user stories ; • Découper les user stories sous forme de tâches à accomplir. Une semaine comporte deux étapes distinctes. La première consiste en la rédaction de tests automatisés qui seront exécutés une fois que l'implémentation sera effectuée. La seconde, quant à elle, consiste en la réalisation proprement dite des user stories. L'objectif est de mettre en place un incrément, c'est à dire une version fonctionnelle et utilisable des fonctionnalités ou des besoins décrits au sein des user stories pour la fin de la semaine. Ce cycle court favorise le retour d'informations rapides et continues.
Cycle trimestriel **(Quarterly Cycle)**	Cette pratique consiste à déterminer et planifier les thèmes à réaliser sur base trimestrielle. Dans ce but, une réunion a lieu une fois tous les trois mois avec pour objectif : • Identifier les problèmes éventuels ; • Déterminer les thèmes à traiter lors du prochain trimestre ; • Sélectionner les user stories à réaliser dans le cadre des thèmes retenus ; • Se concentrer sur le projet dans un plan plus global.

Du mou **(Slack)**	Cette pratique consiste à inclure un certain nombre de tâches mineures pouvant être abandonnées ou postposées au sein d'une itération au cas où l'équipe se retrouverait à court de temps. L'objectif est de garder une certaine marge de manœuvre quand tout ne se déroule pas comme prévu.
Construction en dix minutes **(Ten-Minute Build)**	La compilation automatique du système tout entier et l'exécution des tests pour l'ensemble de celui-ci doivent pouvoir être effectuées en l'espace de dix minutes.
Intégration continue **(Continuous Integration)**	Les changements au niveau du code doivent être intégrés et testés de manière fréquente et continue. L'ordre de fréquence est généralement de quelques heures.
Tester d'abord **(Test-First Programming)**	Les tests unitaires sont rédigés avant l'implémentation de la fonctionnalité et sont exécutés de façon continue.
Conception incrémentale **(Incremental Design)**	Le système est construit de manière incrémentale, c'est-à-dire, pièce par pièce, jour après jour. La conception est réalisée de la manière la plus simple possible. Le code est remanié, ultérieurement, si cela s'avère nécessaire.

2.2.5 Artéfacts

La littérature au sujet de l'extreme programming indique le code comme étant le seul artéfact ayant une réelle valeur intrinsèque. Toutefois, nous ne nous limitons pas ici à cet unique élément, mais nous présentons également les principaux artéfacts considérés comme périphériques. Notons que la liste définie ci-dessous n'est pas exhaustive.

Tableau 6 : Les artéfacts de la méthode XP [Wells, 2013 ; Ijab et al, 2004 ; IBM Corporation, 2007]

Artéfact	Description
Build	Un build est une version fonctionnelle et exécutable (d'une partie) du système composée d'un sous-ensemble des fonctionnalités du produit final.
Code de production	Le code de production est le code source du système correspondant à l'implémentation concrète des user stories.
Plan d'itération	Un plan d'itération spécifie pour une itération la liste des user stories à implémenter. Ce plan contient également la liste des tâches nécessaires à la réalisation de ces user stories.
Plan de release	Un plan de release spécifie pour chaque release la liste des user stories à implémenter. La liste des user stories est triée par ordre de priorité.
Test unitaire	Un test unitaire est une pièce de code destinée à vérifier et valider le bon fonctionnement d'une partie déterminée du système.
Test d'acceptation	Un test d'acceptation est un test fonctionnel destiné à vérifier et valider le comportement d'une fonctionnalité du système. Ce type de test est spécifié par le client à partir des user stories et sert à garantir que le système implémenté répond aux exigences.
User story	Une user story est une description non détaillée et sous forme de langage naturel d'une fonctionnalité requise du système. La description attendue doit être claire et concise. Généralement, les user stories sont rédigées sous la forme de fiches cartonnées ou de Post-It.
Vision	La vision définit l'essence du produit du point de vue des parties prenantes.

2.3 La méthode « Open Unified Process »

2.3.1 Présentation

La méthode Open Unified Process, également connue sous son diminutif OpenUP, est une version adaptée et allégée de RUP (Rational Unified Process). Certains éléments de RUP ont été modifiés, combinés, voire supprimés pour composer cette méthode. Néanmoins, celle-ci conserve ses caractéristiques principales telles que la gestion du risque, le développement à partir de cas d'utilisation et de scénarios, ou encore, une approche centrée sur l'architecture.

OpenUP est décrite de la manière suivante au sein du Framework Eclipse Process Framework (EPF) [Epf.eclipse.org, 2012] :

« *OpenUP is a lean Unified Process that applies iterative and incremental approaches within a structured lifecycle. OpenUP embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. It is a tools-agnostic, low-ceremony process that can be extended to address a broad variety of project types.* »

2.3.2 Auteur(s)

La méthode OpenUP a été définie par le groupe IBM. Aujourd'hui, celle-ci fait partie intégrante du Framework EPF (Eclipse Process Framework) géré et maintenu par la fondation Eclipse.

2.3.3 Rôles et responsabilités

La méthode OpenUP répertorie au total quinze rôles scindés en trois catégories, soit, les rôles de base, les rôles spécifiques liés au déploiement ou encore les rôles spécifiques liés à l'environnement. Nous nous focaliserons ici sur la description des rôles de base.

Tableau 7 : Les rôles et responsabilités de la méthode OpenUP [Epf.eclipse.org, 2012]

Rôle	Responsabilités
Analyste	• Description détaillée des exigences relatives à l'ensemble du système ; • Description détaillée des scénarios de cas d'utilisation ; • Définition de la vision du futur système ; • Identification et spécification des exigences fonctionnelles et non-fonctionnelles du système.

Architecte	• Analyse des besoins et identification des contraintes architecturales ; • Description de l'architecture ainsi que des choix et décisions relatives à celle-ci.
Développeur	• Conception et implémentation (d'une partie) du système ; • Mise en place et validation de l'infrastructure ; • Intégration des modifications et création des packages de déploiement ; • Implémentation et exécution des tests unitaires.
Chef de projet	• Planification du projet ; • Planification des itérations ; • Gestion des itérations ; • Evaluation des résultats.
Parties prenantes	• Responsabilités variables en fonction du groupe d'intérêts.
Testeur	• Création, implémentation et exécution des tests fonctionnels ; • Interprétation et communication du résultat des tests exécutés.
Rôle quelconque	• Détection et introduction des demandes de changement ; • Exécution de tâches d'ordre général.

2.3.4 Pratiques

La méthode OpenUP repose sur un ensemble de pratiques. Celles-ci sont décrites ci-dessous.

Tableau 8 : Les pratiques de la méthode OpenUp [Epf.eclipse.org, 2012 ; Agile Alliance, 2015]

Pratique	Description
Développement itératif **(Iterative Development)**	Le développement itératif consiste à découper un projet en une série d'itérations. Une itération correspond à un laps de temps fixe et précis, durant lequel une série d'activités est entreprise afin de produire ou revoir une partie définie du système.
Release Planning	La pratique de release planning consiste à planifier le projet de manière itérative. Le plan du projet est effectué selon deux niveaux. Tout d'abord, on réalise un planning général à propos de l'entièreté du projet. Ensuite, à partir de ce plan général, on réalise un plan plus détaillé pour chaque itération du projet.
Equipe complète **(Whole Team)**	L'équipe est pluridisciplinaire, auto-organisée, fluide et collaborative. Celle-ci réunit l'ensemble des compétences nécessaires au bon déroulement du projet et s'organise de par elle-même. L'équipe n'est pas fixe. En effet, sa composition peut évoluer au cours du projet selon les profils requis pour atteindre l'objectif défini. Dans un but d'efficacité et d'efficience, la collaboration et la communication sont fortement mises en avant.
Gestion du changement d'équipe **(Team Change Management)**	La gestion du changement fait partie intégrante de la gestion de projet. Les demandes de changement et de résolution de bugs peuvent être initiés à tout moment et par n'importe quel membre de l'équipe. Celles-ci sont ajoutées à la liste des tâches en cours du projet et sont traitées suivant le même processus.
Adaptation du processus sur mesure **(Project Process Tailoring)**	Le processus est adapté et les outils sont sélectionnés en fonction du projet. La complexité, le type ou encore la taille du projet sont des facteurs clés quant aux choix et adaptations effectués.

Contrôle simultané **(Concurrent Testing)**	Les tests sont effectués simultanément au développement lors d'une itération.
Intégration continue **(Continuous Integration)**	Les changements au niveau du code doivent être intégrés et testés de manière fréquente et continue. L'ordre de fréquence est généralement de quelques heures.
Architecture évolutive **(Evolutionary Architecture)**	La conception de l'architecture du système est définie et améliorée de manière itérative et incrémentale. Les principaux problèmes techniques liés à la solution sont analysés et fixés au moment opportun. Toutes les décisions clés et points en suspens concernant l'architecture sont documentés.
Conception évolutive **(Evolutionary Design)**	La conception du système est définie, revue et remodelée au fur et à mesure du développement et ceci de manière continue plutôt que complète et unique en début de projet ou d'itération.
Vision partagée **(Shared Vision)**	Cette pratique a pour objectif de définir, maintenir et communiquer la vision du projet.
Développement piloté par les tests **(Test Driven Development)**	Le développement dirigé par les tests désigne une approche de développement logiciel qui recommande de rédiger les tests unitaires avant de rédiger le code source d'une partie d'une application. D'abord, le développeur écrit un premier test unitaire et vérifie que celui-ci échoue comme attendu (le code source relatif à cet aspect de l'application n'existant pas à cette étape). Ensuite, il rédige le code nécessaire et suffisant pour passer ce dernier avec succès. Finalement, il remanie le code source afin d'en améliorer la qualité.
Développement piloté par les cas d'utilisation **(Use Case Driven Development)**	Le développement dirigé par les cas d'utilisation est une approche de développement logiciel qui consiste à capturer les exigences fonctionnelles sous forme de cas d'utilisation et de scénarios représentant l'interaction entre l'utilisateur et le système. Ces cas d'utilisation et scénarios servent ensuite de fil conducteur pour toutes les activités principales du projet.

2.3.5 Artéfacts

OpenUp propose plus d'une vingtaine d'artéfacts au sein de son Framework. Nous présentons ci-dessous un sous-ensemble de ceux-ci. L'échantillon sélectionné correspond à la liste des principaux artéfacts associés aux rôles de base énoncés précédemment.

Tableau 9 : Les artéfacts de la méthode OpenUp [Epf.eclipse.org, 2012]

Artéfact	Description
Cahier d'architecture	Description de l'analyse réalisée, des hypothèses émises, des problèmes rencontrés, des choix effectués, ainsi que des décisions prises quant à l'élaboration de l'architecture.
Build	Version fonctionnelle et exécutable (d'une partie) du système composée d'un sous-ensemble des fonctionnalités du produit final.
Design	Description technique des éléments et fonctionnalité du système.
Test unitaire	Pièce de code destinée à vérifier et valider le bon fonctionnement d'une partie déterminée du système.
Glossaire	Liste des termes importants associés à leurs définitions utilisés au sein du projet.
Plan d'itération	Plan détaillé d'une itération spécifiant les objectifs de l'itération, un plan de réalisation et de répartition des tâches nécessaires à la réalisation de ceux-ci, ainsi que les critères d'évaluations permettant de valider les éléments produits.
Plan de projet	Plan général et stratégique du projet contenant toutes les informations utiles et nécessaires à sa bonne gestion. On y retrouve par exemple la liste des itérations prévues et leurs objectifs ou encore les différentes pratiques à adopter au sein du projet.

Liste des risques	Liste priorisée de tous les risques afférant au projet. On associe à chaque risque des mesures à prendre afin d'en réduire la probabilité et/ou l'impact, ainsi que des actions à entreprendre au cas où le risque se réalise.
Analyse des besoins à l'échelle du système	Spécification des exigences fonctionnelles et non-fonctionnelles portant sur l'ensemble du système.
Cas de test	Documentation nécessaire à la vérification et validation du bon fonctionnement d'une partie déterminée du système. Un cas de test décrit un scénario d'exécution détaillé et spécifie les données d'inputs, les conditions d'exécutions et les résultats attendus permettant de valider le comportement de la fonctionnalité vérifiée par le scénario décrit.
Rapport de test	Compte-rendu détaillé de l'exécution d'un test ou d'un ensemble de tests. On retrouve dans ce registre tous les détails pertinents quant à l'exécution de ceux-ci. Généralement on y inscrit, entre autre, la date d'exécution et le résultat observé.
Script de test	Collection d'instructions composant un test. Les instructions sont décrites pas à pas. Celles-ci peuvent être décrites textuellement et exécutées, soit manuellement, soit de manière automatisée.
Cas d'utilisation	Description du comportement attendu du système. Chaque cas d'utilisation contient un ou plusieurs scénarios définissant comment le système doit interagir avec les utilisateurs pour atteindre un but.
Vision	Définition de l'essence du produit du point de vue des parties prenantes. En d'autres mots, la vision décrit l'idée et les besoins fondamentaux du produit du point de vue des parties prenantes.
Liste des tâches	Liste complète reprenant tout le travail nécessaire à la réalisation du produit final ou impactant de manière quelconque celui-ci. On retrouve sur cette liste les diverses demandes réalisées au travers des use cases, des demandes de changement ou d'amélioration, …

2.4 D'autres méthodes agiles

Il existe davantage de méthodes agiles que celles citées précédemment. Ces dernières sont différentes en terme de champ d'action, de plan de mise en œuvre, de terminologie ou encore de recommandations. Citons quelques exemples :

- Adaptive Software Development [Highsmith, 2000]
- Crystal clear [Cockburn, 2005]
- Dynamic Systems Development Method [Stapleton, 1997]
- Feature-Driven Development [Palmer et al, 2002]
- …

Troisième étape : Réalisation d'un prototype de référentiel web des méthodes et pratiques agiles

3 Réalisation d'un prototype d'un référentiel web

.

3.1 Généralités

3.1.1 Qu'est-ce qu'un référentiel ?

Le terme référentiel possède diverses significations dans la langue française, celles-ci varient en fonction du contexte et du domaine d'application.

Dans le cas présent, un référentiel correspond à une collection d'objets sur un thème donné.

Dans notre contexte, on s'intéresse aux composants méthodologiques tels que les pratiques.

3.1.2 Les intérêts d'un référentiel

Une terminologie et une sémantique commune (formalisation)

Un référentiel est un outil qui favorise l'adoption et le partage d'une terminologie et d'une sémantique commune.

En effet, celui-ci permet de consulter facilement et rapidement les différents termes appartenant à la terminologie d'un domaine défini, ainsi que de disposer d'une explication fiable de ces derniers.

De ce fait, un référentiel est d'un intérêt particulier dans le cadre de l'adoption des méthodes agiles où l'emploi d'une nouvelle méthode de travail implique l'utilisation d'une terminologie spécifique propre à cette dernière.

Notons que la communication est l'un des éléments clé pour mener à bien un projet. Or, il est parfois difficile de communiquer sur un sujet donné, surtout si le contexte de ce dernier n'est pas familier pour tous les interlocuteurs. L'emploi d'un terme incohérent, non-approprié ou inconnu de certains pour désigner un concept est, généralement, la source de malentendus et finit par nuire à l'efficacité. A contrario, l'utilisation d'un langage commun, aide quant à lui, à mieux se comprendre, à mieux communiquer et à mieux échanger.

Une mine d'or d'informations

De par son contenu et sa structure, un référentiel s'avère être une mine d'or d'informations. Celui-ci tente de couvrir un thème donné de manière optimale et propose des informations de références de façon structurée et centralisée. Dès lors, il est beaucoup plus simple de parvenir à accéder à une information visée. De plus, le coût de la recherche est minime, et le coût de l'échec de la recherche est fortement limité au sein d'un référentiel.

Notons qu'en ce qui concerne les méthodes et pratiques agiles, il est fort de constater l'existence de nombreux ouvrages, articles ou encore sites web dédiés à ce sujet.

Toutefois, il est souvent laborieux de collecter de l'information ciblée et complète à ce propos. En effet, entre les ouvrages payants, les documentations axées sur une méthode unique, les articles quelquefois peu objectifs ou encore les données sur la toile parfois disparates ou peu fiables, il faut faire preuve de patience et de persévérance afin de rassembler les informations souhaitées.

Une boîte à outils

Un référentiel s'avère être une boîte à outils fantastique. Celui-ci correspond à un recueil de bonnes pratiques reconnues sur lequel on peut s'appuyer.

Nous mettons, ici, l'accent sur l'utilité d'une telle boite à outils dans le cadre de l'adoption des méthodes agiles. En effet, avoir en sa possession une panoplie d'outils éprouvés comme base, est un atout indéniable permettant une avancée rapide et une limitation du risque d'échecs.

3.1.3 Les qualités primordiales d'un référentiel

Les qualités primordiales d'un référentiel sont :

- La définition des termes d'un référentiel doit être simple à comprendre, ce qui suppose l'emploi d'un vocabulaire adapté à l'audience ;
- La définition des termes d'un référentiel doit être objective et ne doit pas laisser place à l'interprétation ;
- La définition des termes d'un référentiel doit être cohérente afin qu'il soit possible de choisir entre deux termes sans difficulté ;
- Le contenu d'un référentiel doit être complet et minimal. Ce dernier doit couvrir le domaine métier de manière exhaustive, mais doit être réduit au minimum nécessaire pour y parvenir ;
- Un référentiel doit être extensible. Cela signifie qu'il doit pouvoir accueillir de nouveaux termes sans en compromettre son équilibre global ;
- Un référentiel doit être révisable. Cela signifie que son contenu doit pouvoir être modifié lorsque certains termes doivent être adaptés, fusionnés ou scindés, et ce, sans compromette son équilibre global.

3.1.4 Les principaux référentiels existants sur le thème de l'agilité

Alliance Agile – Guide des pratiques agiles

Présentation :

Le guide des pratiques agiles de l'Alliance Agile[2] est certainement le référentiel en ligne anglophone le plus connu sur le thème de l'agilité. Au sein de celui-ci, les différents termes pouvant être consultés sont présentés soit sous la forme d'une liste triée par ordre alphabétique, soit sous la forme d'une carte de métro (Figure 13).

Le canevas de description des pratiques répertoriées est le suivant :

- Le nom ;
- La description ;
- Les synonymes parmi la terminologie agile ;
- Les erreurs courantes de mise en œuvre ;
- Les signes observables de l'utilisation dans l'espace de travail ;
- Les bénéfices attendus ;
- Le contexte historique présentant les origines et l'évolution ;
- Les références vers des travaux scientifiques.

Critique :

Le canevas proposé pour la description des pratiques est intéressant. En effet, la trame définie est plutôt complète et judicieuse. Néanmoins, on peut regretter le manque d'assiduité quant à la complétion de celle-ci pour de nombreux éléments du référentiel, ainsi que l'absence de certains éléments au niveau du canevas tels que les objectifs d'amélioration de la pratique, ou encore des recommandations quant à l'adoption de la pratique.

Ensuite, la présentation des termes du référentiel sous la forme d'une carte de métro, où chaque rail correspond à une catégorie, apporte une dimension intéressante. Celle-ci permet de catégoriser les pratiques selon différents domaines de préoccupations. Toutefois, la catégorisation est assez minimaliste et manque d'uniformité.

D'un point de vue ergonomique, on note une présentation sobre et soignée. Cependant, le site est peu adapté pour les appareils mobiles. Certes, le contenu reste lisible sur la plupart des écrans de petite taille, par contre, la navigation et la sélection des éléments sont fort peu adaptées, surtout en mode portrait. Toujours du point de vue de

[2] http://guide.agilealliance.org

l'ergonomie, nous regrettons le fait de ne pas pouvoir filtrer la liste des pratiques proposées ou encore de rechercher une pratique particulière.

Finalement, bien que la carte de métro regroupe certaines pratiques sous la coupe des méthodes Scrum, XP et Lean, le contenu du référentiel se focalise sur les pratiques agiles. La notion de méthode et des concepts s'y afférant tels que les rôles et les artéfacts sont absents. Des notions pourtant fortes intéressantes dans le contexte de l'adoption des méthodes agiles.

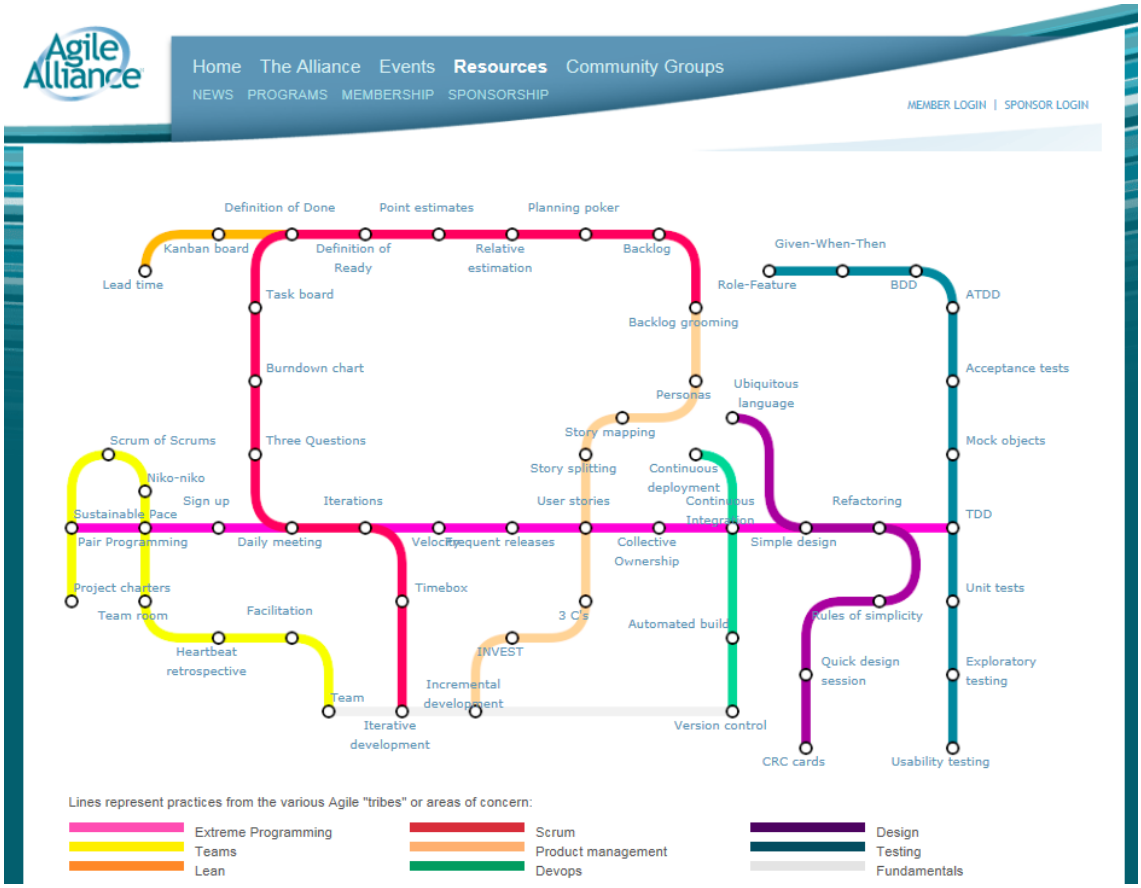


Figure 13 : Alliance Agile - Guide des pratiques agiles

Institut Agile – Référentiel des concepts, pratiques et compétences agiles

Présentation :

Le référentiel des concepts, pratiques et compétences agiles de l'Institut Agile[3] est l'équivalent francophone du guide des pratiques agiles de l'Alliance Agile. Au sein de celui-ci, les différents termes peuvent être triés soit par ordre alphabétique, soit par type (Compétence, Concept, Pratique).

Le canevas de description des termes répertoriés est similaire à celui-ci de son homologue anglophone :

- Le nom ;
- La description ;
- Les synonymes parmi la terminologie agile ;
- Les erreurs courantes de mise en œuvre ;
- Les signes observables de l'utilisation dans l'espace de travail ;
- Les bénéfices attendus ;
- Le contexte historique présentant les origines et l'évolution ;
- Les références vers des travaux scientifiques.

Critique :

Comme indiqué pour son homologue, le canevas proposé est intéressant. Néanmoins, on peut regretter le manque d'assiduité quant à la complétion de celui-ci pour de nombreux éléments du référentiel, ainsi que l'absence de certains éléments au niveau du canevas tels que les objectifs d'amélioration de la pratique, ou encore des recommandations quant à l'adoption de la pratique.

Du point de vue de l'ergonomie, nous notons également les mêmes lacunes que le référentiel de l'Alliance Agile.

En ce qui concerne la structure et le contenu, celui-ci se focalise également sur les pratiques.

Toutefois, nous notons la présence de quelques fonctionnalités particulièrement intéressants comme :

- La possibilité de télécharger en format PDF l'intégralité du contenu du référentiel ;
- La mise à disposition d'outils liés au référentiel tels que le détrompeur (*) et le souligneur (**).

[3] http://referentiel.institut-agile.fr

(*) Le détrompeur est un marque-page qui permet aux lecteurs de mettre en évidence les termes définis dans le référentiel sur une page web. Celui-ci donne également la possibilité de consulter aisément les informations relatives à ceux-ci. En pratique, lors de la consultation d'une page web contenant des termes définis au sein du référentiel, ces derniers s'affichent en surbrillance. En survolant un de ces termes, un cadre contenant la description provenant du référentiel apparaît. En cliquant sur le terme, on est redirigé vers la page complète du référentiel.

(**) Le souligneur est un outil semblable au détrompeur mais destiné aux auteurs. Une fois intégré dans le site de l'auteur, il permet d'offrir aux lecteurs de manière automatique et systématique les fonctionnalités proposées par le détrompeur.

Figure 14 : Institut Agile - Référentiel des concepts, pratiques et compétences agiles

3.2 Démarche

Ce travail ayant pour thème les méthodes et pratiques agiles, nous avons souhaité procéder à la réalisation du prototype de manière agile.

De ce fait, nous avons tout d'abord sélectionné une méthodologie à suivre. Notre choix s'est orienté vers Scrum car cette dernière est fort populaire et peu prescriptive. Cependant, le contexte et le cadre de travail étant quelque peu particulier, nous avons dû faire une entorse aux règles de la méthode. En effet, le travail à réaliser étant individuel, nous avons dû assumer personnellement l'ensemble des différents rôles.

Une fois la méthodologie sélectionnée, nous avons abordé la phase d'initiation, où nous avons fixé le cap du projet en formalisant la vision du produit, décrivant les personas, et définissant un product backlog.

Ensuite, nous avons entamé une phase de préparation. Durant celle-ci, nous avons effectué les choix technologiques, installé et configuré l'environnement de travail et défini l'architecture globale de l'application afin de mettre en place des bases solides pour la suite du projet et pouvoir démarrer les sprints dans de bonnes conditions.

Une fois la phase de préparation terminée, nous avons entrepris une série de six sprints de trois semaines afin de réaliser le prototype.

Finalement, nous avons entrepris la phase de clôture visant à préparer le produit pour sa livraison.

3.3 Analyse et spécification du prototype

3.3.1 Définition de la vision du produit

La vision du produit a pour but de décrire l'essence même de ce dernier, son objectif central, à qui il s'adresse, ainsi que la finalité recherchée au travers de sa réalisation.

La vision doit être formulée de manière claire, engageante et concise. Afin de s'assurer que celle-ci réponde à ces critères, on procède généralement au test dit de l'ascenseur (« elevator pitch »). Il consiste à pouvoir expliquer le produit endéans le laps de temps nécessaire pour monter avec un ascenseur.

De ce fait, nous avons décrit la vision de notre produit en respectant le canevas suivant :

Pour	(le public cible)
Qui	(les besoins du public cible)
Le	(le nom du produit)
Est un	(la catégorie du produit)
Qui	(les avantages principaux du produit)
A la différence de	(les concurrents principaux)
Notre produit	(les éléments de différentiation principaux)

La vision du produit est ici la suivante :

*« **Pour** les experts en méthodologie, **qui** souhaitent documenter les méthodes et pratiques agiles existantes, **ainsi que** pour toutes les personnes ayant un intérêt envers celles-ci et **qui** souhaitent s'informer sur le sujet, **le** référentiel web Agilia **est un** référentiel web **qui** offre un contenu riche et structuré sur le thème de l'agilité.*

***A la différence** d'autres référentiels sur ce même thème, **notre produit** aborde non seulement le concept de pratique, mais également de méthode. A terme, celui-ci abordera aussi différents autres concepts afférents aux méthodes tels que les rôles et artéfacts. De plus, celui-ci offre une expérience utilisateur optimale grâce à une mise en page dite « adaptative » permettant d'ajuster automatiquement le contenu des pages de l'application à la largeur de l'écran, quel que soit l'appareil utilisé. »*

3.3.2 Profils utilisateur : Définition des personas

Afin de pouvoir concevoir un produit adapté aux besoins des utilisateurs et offrir une expérience utilisateur optimale à ces derniers, nous avons identifié les types d' utilisateurs cibles du référentiel et définis ceux-ci sous forme de personas.

Les personas sont des personnages fictifs créés de toutes pièces permettant de représenter les types d'utilisateurs cibles d'un produit. La création des personas a pour but de mieux comprendre le comportement de ces derniers, et ainsi à orienter la conception du produit en fonction de leurs besoins spécifiques, leurs préoccupations et leurs comportements.

1. Jamie – L'expert

Jamie
L'expert

51 ans
Professeur d'université

Background

Jamie possède un doctorat en sciences économiques et de gestion. Il est professeur d'université dans un prestigieux établissement où il enseigne les méthodes modernes de gestion de projet. Jamie est un véritable expert en ce qui concerne la méthodologie agile. Il est notamment l'auteur de nombreux ouvrages portant sur le sujet. Il détient également diverses certifications telles que « PMI Agile Certified Practictioner » et « Certified Scrum Coach ». En dehors de ses activités à l'université, Jamie accompagne et coache des entreprises dans leur processus de transition à l'agilité.

Ses caractéristiques

Jamie a l'esprit cartésien. Il est rigoureux, passionné et enthousiaste. Ses hobbies sont la lecture et les jeux d'échec en ligne. Jamie est peu friand des appareils mobiles. Pour lui, rien de tel qu'un ordinateur de bureau. Il aime se rendre utile et partager son savoir. De par son métier, Jamie aime que les choses soient claires et faciles à comprendre.

Sa devise

L'information est l'oxygène des temps modernes.

Ses objectifs

Jamie souhaite partager son savoir et promouvoir la méthodologie agile. Pour y parvenir, il aimerait entre autre documenter et publier en ligne des informations à propos des méthodes et pratiques agiles.

Figure 15 : Persona - Jamie L'expert

2. Fred – Le novice

Fred
Le novice

23 ans
Analyste-Programmeur

Background

Fred est un jeune diplômé en sciences de l'information. Il travaille depuis peu en tant qu'analyste-programmeur au sein d'une grande entreprise qui a récemment adopté l'approche agile dans le cadre de ses activités de digitalisation. Bien que la notion d'agilité ait été évoquée au cours de son cursus scolaire, Fred est novice en la matière.

Ses caractéristiques

Fred est passionné de nouvelles technologies. Il possède quasiment tous les types d'appareils de dernière génération (ex : Smartphone, Tablette, etc.). Il jongle avec ceux-ci quotidiennement. Il ne se sépare jamais de son smartphone et reste constamment connecté. Il est avide de connaissance. Il ne supporte pas rester dans l'ignorance et veut toujours apprendre de nouvelles choses. Dès lors, il « google » tout ce qui est flou ou inconnu pour lui.

Notons que Fred est peu patient, rien ne l'excède davantage que de devoir effectuer de multiples recherches afin de trouver une information.

Sa devise

Google est ton ami !

Ses objectifs

Fred a pour dessein de se renseigner sur la méthodologie agile, ainsi que se familiariser avec la terminologie. Pour cela, il désire obtenir un aperçu des méthodes et pratiques qui existent et aborder les détails de celles-ci. De plus, Fred souhaite accéder à des informations de confiance, centralisées, et qui soient consultables de manière optimale à partir de ses différents appareils.

Figure 16 : Persona - Fred Le novice

3. Martin – Le pratiquant

Martin
Le pratiquant

41 ans
Chef de projet

Background

Martin a une formation d'ingénieur. Il travaille depuis une dizaine d'années en tant que chef de projet informatique pour une institution financière. Depuis environ deux ans, Martin réalise des projets qu'il dirige avec brio. Martin n'est, certes, pas un expert en ce qui concerne la méthodologie agile. Néanmoins, il possède un certain niveau d'expérience théorique et pratique. L'entreprise qui l'emploie lui a demandé de tester la méthode agile dans le cadre de ses nouveaux projets.

Ses caractéristiques

Martin est perfectionniste. Il est le meilleur pour diriger et motiver une équipe. Malheureusement, il n'est pas de nature aventureuse. Il préfère ne pas prendre trop de risques et travailler avec des systèmes bien rodés.

Sa devise

Un pour tous, tous pour un !

Ses objectifs

Martin a pour dessein de se renseigner sur les différentes méthodes et pratiques agiles existantes, mais également donner son avis, ainsi que partager des retours d'expérience à propos de ces dernières.

Figure 17 : Persona - Martin Le pratiquant

3.3.3 Spécification des exigences fonctionnelles : Définition du product backlog

Le product backlog spécifie et priorise l'ensemble des fonctionnalités du produit que l'on veut développer. L'objectif est d'avoir une vue globale sur le travail à entreprendre et d'implémenter en premier ce qui a le plus de valeur.

Tableau 10 : Product backlog du prototype

ID	Elément	Estimation (md)	Priorité	Statut
1	Créer un compte utilisateur	2	Très haute	Réalisé
2	S'authentifier	1	Très haute	Réalisé
3	Se déconnecter	0,5	Très haute	Réalisé
4	Créer une fiche de type pratique	1,5	Très haute	Réalisé
5	Consulter la liste des fiches de type pratique triées par ordre alphabétique	1	Très haute	Réalisé
6	Consulter les informations détaillées d'une fiche de type pratique	1	Très haute	Réalisé
7	Supprimer une fiche de type pratique	1	Très haute	Réalisé
8	Editer une fiche de type pratique	3	Très haute	Réalisé
9	Créer une fiche de type méthode	1,5	Haute	Réalisé
10	Consulter la liste des fiches de type méthode triées par ordre alphabétique	1	Haute	Réalisé
11	Consulter les informations détaillées d'une fiche de type méthode	1	Haute	Réalisé
12	Supprimer une fiche de type méthode	1	Haute	Réalisé

13	Editer une fiche de type méthode	3	Haute	Réalisé
14	Associer des fiches de type pratique à une fiche de type méthode	3	Haute	Réalisé
15	Consulter la liste des fiches de type pratique associées à une fiche de type méthode	1	Haute	Réalisé
16	Dissocier des fiches de type pratique d'une fiche de type méthode	2	Haute	Réalisé
17	Filtrer les éléments de la liste de fiches de type pratique selon le contenu d'une zone de texte libre	0,5	Moyenne	Réalisé
18	Filtrer les éléments de la liste de fiches de type méthode selon le contenu d'une zone de texte libre	0,5	Moyenne	Réalisé
19	Commenter une fiche de type pratique	1	Faible	Réalisé
20	Commenter une fiche de type méthode	1	Faible	Réalisé
21	Supprimer un commentaire posté sur une fiche de type pratique	1	Faible	Réalisé
22	Supprimer un commentaire posté sur une fiche de type méthode	1	Faible	Réalisé
23	Modérer à postériori les commentaires postés par les utilisateurs.	0,5	Faible	Réalisé

3.3.4 Définition des rôles utilisateurs

Un référentiel vise à proposer des informations de références, c'est-à-dire des informations fiables. Dès lors, il est primordial de s'assurer que ces dernières ne soient pas altérées par n'importe qui. C'est pourquoi, afin d'assurer une gestion propre, efficace et sécurisée du référentiel, il est impératif de définir des rôles au sein de celui-ci. Un rôle détermine l'identité d'un utilisateur par rapport à ses permissions. Les rôles que nous avons définis sont les suivants :

- Administrateur

  Un administrateur est un utilisateur connecté et authentifié disposant des droits d'administrateur. Un administrateur est au niveau supérieur des catégories de privilèges. Il a un pouvoir complet sur l'application.

  Ce rôle est attribué au propriétaire de l'application.

- Editeur

  Un éditeur est un utilisateur connecté et authentifié disposant des droits d'éditeur. Un éditeur peut consulter le contenu du référentiel, éditer ce dernier, et poster des commentaires à son propos.

  Ce rôle est attribué par l'administrateur à un nombre limité d'utilisateurs reconnus en tant qu'experts en ce qui concerne la méthodologie agile.

- Visiteur authentifié

  Un visiteur authentifié est un utilisateur connecté et authentifié ne disposant pas de droits particuliers. Celui-ci peut consulter le contenu du référentiel et poster des commentaires à son propos.

  Ce rôle est attribué par défaut à tout nouvel utilisateur lors de son inscription.

- Visiteur anonyme

  Un visiteur anonyme est un utilisateur qui visite le site sans être connecté et authentifié. Celui-ci dispose toujours d'un accès en "lecture-seule". Il ne peut donc pas éditer le contenu du référentiel, ni même poster de commentaires.

  Ce rôle est attribué de manière automatique à tout utilisateur qui visite le site sans être connecté et authentifié.

3.3.5 Définition des droits et privilèges

Le tableau suivant offre un aperçu détaillé des droits et privilèges que nous avons définis et attribués aux utilisateurs du référentiel en fonction de leur rôle.

Tableau 11 : Aperçu des droits et privilèges au sein du référentiel

Privilège	Rôle			
	Visiteur anonyme	**Visiteur authentifié**	**Editeur**	**Admin**
Consulter l'aperçu des fiches de type méthode	X	X	X	X
Consulter l'aperçu des fiches de type pratique	X	X	X	X
Filtrer l'aperçu des fiches de type méthode	X	X	X	X
Filtrer l'aperçu des fiches de type pratique	X	X	X	X
Consulter les détails d'une fiche de type méthode	X	X	X	X
Consulter les détails d'une fiche de type pratique	X	X	X	X
Créer une fiche de type méthode			X	X
Créer une fiche de type pratique			X	X
Editer une fiche de type méthode			X	X
Editer une fiche de type pratique			X	X
Supprimer une fiche de type méthode			X	X
Supprimer une fiche de type pratique			X	X
Associer des fiches de type pratique à une fiche de type méthode			X	X
Dissocier des fiches de type pratique d'une fiche de type méthode			X	X
Poster un commentaire		X	X	X
Supprimer un commentaire		X	X	X
Modérer les commentaires				X
Créer un compte utilisateur	X			
S'authentifier	X			
Se déconnecter		X	X	X

3.3.6 Définition du canevas des fiches du référentiel

Le référentiel contiendra deux types de fiches distinctes. On notera d'un côté les fiches de type pratique, et de l'autre, les fiches de type méthode. Pour chaque type de fiche, nous avons défini un canevas de description détaillée à partir d'éléments qui nous ont semblé pertinents.

Les canevas définis sont les suivants :

Tableau 12 : Canevas d'une fiche de type pratique du référentiel

Eléments	Question à se poser
Nom	Quel nom décrit le mieux la pratique ?
Description	Quelles sont les informations essentielles qui permettent de décrire la pratique ?
Objectifs d'amélioration	Quels sont les principaux objectifs d'amélioration que confèrent l'adoption de la pratique au sein d'un projet (ex : réduction du time) ?
Synonymes	Quels sont les synonymes connus qui désignent la même pratique ou des pratiques tellement voisines que celles-ci peuvent être considérées comme une seule et même pratique ?
Bénéfices	Quels sont les principaux bénéfices apportés par l'utilisation de la pratique ?
Signes observables	Quels signes observables permettent de reconnaître l'utilisation de la pratique au sein d'un espace de travail ?
Erreurs courantes	Quelles sont les principales erreurs courantes rencontrées quant à l'utilisation de la pratique ?
Recommandations pour l'adoption	Quelles sont les principales recommandations liées à l'adoption de la pratique ?
Origines	Qui sont les personnes qui ont élaboré la pratique ? Quelles informations essentielles permettent de retracer les origines et l'évolution de la pratique ?
Références	Quels sont les ouvrages scientifiques les plus pertinents à propos de la pratique ?

Tableau 13 : Canevas d'une fiche de type méthode du référentiel

Eléments	Question à se poser
Nom	Quel nom décrit le mieux la méthode ?
Description	Quelles sont les informations essentielles qui permettent de décrire la méthode ?
Champ d'application	Quel est l'ensemble des situations pour lesquelles l'usage de la méthode est approprié ? Quelles sont les limitations quant à l'usage de la méthode ?
Pratiques	Quelles sont les pratiques qui composent la méthode ?
Recommandations pour l'adoption	Quelles sont les principales recommandations liées à l'adoption de la méthode ?
Origines	Qui sont les personnes qui ont élaboré la pratique ? Quelles informations essentielles permettent de retracer les origines et l'évolution de la méthode ?
Références	Quels sont les ouvrages scientifiques les plus pertinents à propos de la méthode ?

3.3.7 Définition de l'architecture globale

Toute application web développée avec un langage orienté-objet et qui se veut évolutive se doit d'être rigoureusement architecturée. De ce fait, nous avons porté une attention particulière à la définition de l'architecture du prototype. La Figure 18 schématise l'architecture globale de ce dernier. Celle-ci correspond à une combinaison de l'architecture multicouche et MVC (Modèle - Vue - Contrôleur).

Architecture multicouche

L'architecture du prototype est une architecture 3-tiers, également appelée architecture à trois niveaux.

Le principe consiste à découper l'application en trois niveaux d'abstraction, c'est-à-dire en trois couches distinctes et indépendantes l'une de l'autre :

- La couche de présentation gère l'affichage des données et les interactions entre l'utilisateur et l'application ;

- La couche métier correspond à la partie fonctionnelle de l'application. Celle-ci contient la logique applicative et décrit les opérations que l'application opère sur les données ;

- La couche d'accès aux données gère le stockage et l'accès aux données persistantes du système.

Cette découpe a pour objectif principal de répartir les rôles et de séparer les traitements parmi les couches définies, ainsi que de réduire les dépendances entre celles-ci afin d'accroître la flexibilité, la maintenabilité, ou encore la sécurité de l'application.

Notons que chaque couche regroupe des composants ayant une forte cohésion et ne peut communiquer qu'avec la strate se situant en aval. Ainsi, la couche de présentation ne peut communiquer qu'avec la couche métier, et la couche métier ne peut communiquer qu'avec la couche d'accès aux données.

De plus, la couche métier et la couche d'accès aux données du prototype possèdent chacune des interfaces. De ce fait, la couche supérieure ne connait de celle-ci que ses interfaces et non son implémentation. Ceci permet d'assurer l'indépendance entre les différentes strates de l'application.

Le choix de ce type d'architecture a été motivé par les avantages que procure l'adoption d'un tel patron d'architecture :

- Evolutivité, flexibilité et maintenabilité élevée ;
- Facilité de déploiement ;
- Modularité ;
- Réutilisabilité des composants ;
- Sécurité accrue.

Architecture MVC

En complément, nous avons implémenté le patron de conception MVC (Modèle – Vue – Contrôleur) au niveau de la couche de présentation de l'application afin d'organiser de manière logique et rigoureuse le code en séparant les données, la présentation et la logique de contrôle.

La mise en place de ce patron de conception a été réalisée à l'aide du framework Spring MVC. Dans ce dernier, les contrôleurs sont chargés d'appeler les méthodes nécessaires de la couche métier afin de transmettre à la vue le modèle contenant les données requises pour la génération de celle-ci.

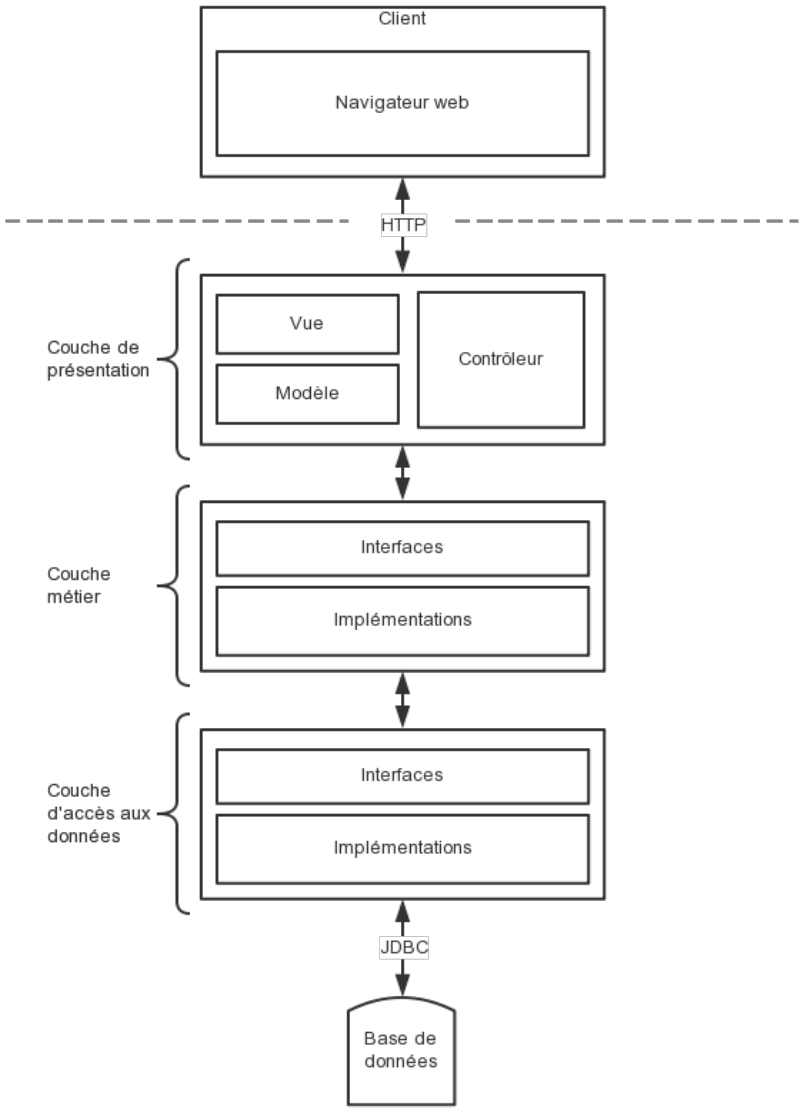


Figure 18 : Architecture globale du prototype

3.3.8 Interface Homme Machine

Une des caractéristiques principales d'un référentiel est un taux de consultation élevé. Dès lors, une interface homme machine simple, claire et intuitive est nécessaire.

De plus, le comportement des utilisateurs a fortement évolué ces dernières années suite à l'avènement des appareils mobiles. Aujourd'hui, les utilisateurs peuvent surfer de n'importe où, n'importe quand, et à partir de différents types d'appareils [Daniel et al, 2013]. Cela est certes intéressant, mais faut-il encore que le site web visité soit adapté à l'écran sur lequel il est visualisé.

C'est pourquoi, une interface homme machine adaptative est souhaitée afin d'offrir une expérience utilisateur optimale quel que soit le terminal utilisé. Par adaptative, on entend une interface fluide et dynamique conçue de manière à pouvoir s'adapter au terminal utilisé [Granddictionnaire, 2015]. Notons que les principaux types de terminaux sont les smartphones, les tablettes et les ordinateurs.

3.4 Conception du prototype

3.4.1 Les langages de programmation

Le prototype a été développé à l'aide de plusieurs langages de programmation.

Tout d'abord, la partie back-end de l'application a été réalisée à l'aide du langage Java. Ensuite, la partie front-end a quant à elle été implémentée en suivant les standards du web, c'est-à-dire, avec les langages HTML5 pour la création des pages web, CSS3 pour la mise en forme de celles-ci, et JavaScript pour le côté dynamique de ces dernières. Finalement, le langage JSP nous a permis d'effectuer l'interfaçage approprié et nécessaire entre la partie HTML5 et Java.

Notons que le choix de ces langages n'a pas été réalisé de manière anodine. En effet, outre les raisons techniques telles la nécessité d'un langage de programmation orienté objet côté back-end, ou encore la possibilité de réaliser un site web adaptatif pour la partie front-end, ceux-ci sont des langages populaires et éprouvés que nous maitrisons. De plus, nous avons pu constater dans le cadre de nos activités professionnelles que cette combinaison est utilisée dans bon nombre d'entreprises afin de mettre en place des applications web de types et tailles diverses.

3.4.2 L'environnement de développement

Le choix d'un environnement de développement est une étape critique d'un projet. En effet, ce dernier entraînera ou non des gains en temps et en effort, donc en coût de développement. C'est pourquoi, nous avons pris le temps nécessaire pour sélectionner l'outil le plus pertinent répondant aux critères suivants :

- Permettre la génération d'applications web ;
- Permettre la programmation orientée objet ;
- Supporter les langages de programmation choisis ;
- Etre gratuit ou à notre disposition ;
- Etre simple à mettre en place et configurer ;
- Etre connu afin de ne pas générer une charge d'apprentissage supplémentaire importante.

Sur base de ces critères, notre choix initial s'est porté vers l'environnement de développement intégré Eclipse. Toutefois, comme nous souhaitions utiliser le framework Spring, nous avons finalement sélectionné Spring Tool Suite (STS), une version préconfigurée de Eclipse contenant d'emblée les plugins basiques pour faire du Spring.

Notons que ce choix s'est avéré judicieux. En effet, celui-ci nous a épargné pas mal de labeur au niveau de la mise en place et de la configuration de l'IDE.

3.4.3 Le gestionnaire de projets

Nous avons eu recours au gestionnaire de projets open source Apache Maven afin de faciliter et automatiser la gestion, la construction et le déploiement de notre projet.

Notons que nous aurions pu utiliser Ant en lieu et place de Maven. Toutefois, notre choix s'est porté sur ce dernier pour sa facilité de prise en main et sa simplicité d'utilisation. De plus, celui-ci était disponible par défaut dans l'environnement de développement utilisé.

3.4.4 Le système de gestion de base de données

Le système de gestion de base de données que nous avons utilisé est le SGBD relationnel MySQL. Celui-ci a été choisi pour sa facilité de déploiement et de prise en main, sa fiabilité et sa rapidité, sa portabilité, sa gratuité, mais également pour les outils graphiques de gestion et de conception disponibles tels que MySQL Workbench et Toad For MySQL.

Notons que le rejet de système de gestion de base de données comme Oracle est fondé sur le principe que notre prototype ne nécessite pas une technologie aussi « lourde » et « avancée ».

3.4.5 Le framework Bootstrap

Afin de démarrer sur des bases solides et de faciliter le développement des interfaces utilisateurs de notre application web adaptative, nous avons recouru à un framework HTML, CSS et JavaScript.

Parmi ce type de framework, de nombreuses solutions s'offraient à nous telles que Bootstrap, Foundation, Gumby, HTML Kickstart, Kube et Pure.

Après une étude comparative et quelques tests sur ces différents frameworks, notre choix s'est porté sur le framework Bootstrap pour les raisons suivantes :

- Disponibilité de nombreux plugins ;
- Communauté active importante ;
- Popularité du framework ;
- Simplicité d'utilisation ;
- Documentation claire, complète et abondante ;
- Compatibilité avec les différents navigateurs ;

Rétrospectivement, celui-ci semble s'être révélé un choix judicieux. En effet, nous avons noté un gain de temps considérable quant à la mise en place des interfaces utilisateurs suite à l'utilisation de ce framework. Nous mettrons également en évidence la compatibilité avec les différents navigateurs testés et la facilité de mise en place d'un site web adaptatif. Nous tempèrerons tout de même notre engouement en soulignant comme bémol le manque d'originalité des thèmes disponibles par défaut.

3.4.6 Le framework Spring

Nous avons recouru au framework Spring afin de structurer, de simplifier et d'accélérer le développement de notre application.

Spring est un framework open source facilitant la réalisation d'application Java en offrant un soutien au niveau de l'infrastructure globale de cette dernière. Celui-ci est basé sur la notion de conteneur léger, c'est-à-dire une infrastructure similaire à un serveur d'application J2EE. Spring a l'avantage d'être modulaire. Cela signifie qu'il n'est pas nécessaire d'implémenter la totalité du framework Spring pour pouvoir l'utiliser. Ainsi, il est donc possible de recourir uniquement aux modules souhaités en fonction du contexte du projet.

Lors de la réalisation de notre application, nous avons utilisé plusieurs de ces modules. Nous soulignerons en particulier l'utilisation du module Spring Security pour la gestion de la sécurité de l'application (ex : gestion de l'authentification, gestion des autorisations), et Spring MVC pour la mise en place du patron de conception MVC dans la couche de présentation de cette dernière.

Nous avons choisi d'utilisé le framework Spring pour les raisons suivantes :

- Spring est considéré comme un standard dans l'industrie du développement d'applications basé sur la plateforme Java. De nombreuses entreprises ont recours à ce dernier ;

- Spring est modulaire et permet de répondre à une large gamme de besoins ;

- Spring dispose d'une documentation claire, complète et abondante ;

- Spring favorise l'intégration avec de nombreux autres frameworks tels que Hibernate.

De plus, celui-ci étant utilisé au sein de multiples projets dans le cadre de nos activités professionnelles, nous souhaitions d'autant plus en découvrir les rouages et les finesses.

3.4.7 Le framework Hibernate

Nous avons recouru au framework Hibernate pour faciliter le développement de la couche d'accès aux données de notre application.

Hibernate est un framework open source de persistance de type ORM (Object Relational Mapping). Celui-ci permet de faciliter la correspondance entre des données stockées dans des objets Java et une base de données. L'utilisation de ce framework permet, entre autres, de réduire le temps de développement d'une application en évitant l'écriture de code répétitif et d'améliorer la portabilité de ce dernier.

Notons qu'il existe d'autres frameworks de persistance de type ORM, comme MyBatis, JDO et TopLink. Cependant, nous avons choisi d'utiliser le framework Hibernate pour sa popularité, sa facilité de prise en main, sa richesse du point de vue des fonctionnalités proposées, ainsi que ses performances élevées.

3.4.8 Les diverses librairies JavaScript

Coder en JavaScript sans utiliser de librairies tierces est parfois long et éprouvant. De ce fait, nous avons recouru à certaines de ces dernières lors la réalisation du prototype.

Tout d'abord, nous avons tiré parti de la librairie jQuery dans le but de simplifier l'écriture du code JavaScript.

Ensuite, pour d'éviter de réinventer la roue en matière de recherche, tri, et filtrage au niveau des listes et tables HTML, nous avons recouru à la librairie List.js spécialisée dans ce domaine.

3.4.9 L'arborescence de fichiers du projet

L'arborescence de fichiers du projet est illustrée en Figure 17. Celle-ci est conforme à la structure attendue par Maven et correspond à une arborescence de fichiers typique d'une application web.

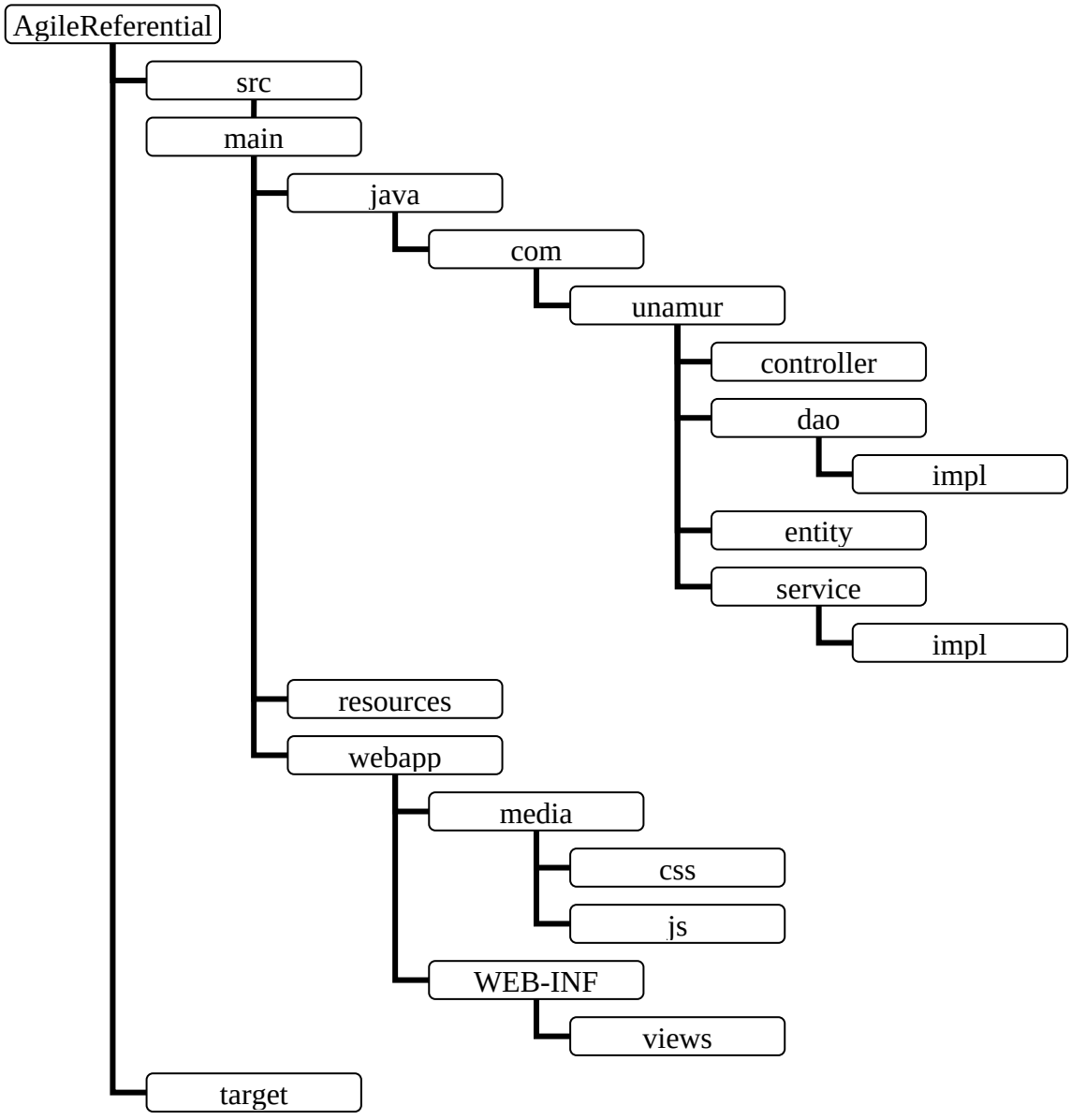


Figure 19 : Arborescence de fichiers du projet

- **AgileReferential/src**

  Ce dossier correspond au répertoire de travail. Celui-ci contient :

  - le sous-dossier *main/java* ;
  - le sous-dossier *main/resources* ;
  - le sous-dossier *main/webapp*.

- **AgileReferential/src/main/java**

  Ce dossier contient les sources java du projet. Celles-ci sont organisées de la manière suivante :

  - le sous-dossier *com/unamur/controller* contient le code source des contrôleurs.
  - le sous-dossier *com/unamur/dao* contient le code source des interfaces de la couche d'accès aux données.
  - le sous-dossier *com/unamur/dao/impl* contient le code source des classes qui implémentent les interfaces de la couche d'accès aux données.
  - le sous-dossier *com/unamur/entity* contient le code source des entités.
  - le sous-dossier *com/unamur/service* contient le code source des interfaces de la couche applicative.
  - le sous-dossier *com/unamur/service/impl* contient le code source des classes qui implémentent les interfaces de la couche applicative.

- **AgileReferential/src/main/resources**

  Ce dossier contient les divers fichiers de propriétés. On y retrouve notamment le fichier de messages par défaut *messages.properties* et le fichier de configuration de log4j *log4j.properties*.

- **AgileReferential/src/main/webapp**

  Ce dossier correspond au répertoire d'application web du projet. Celui-ci contient :

  - le sous-dossier *media* ;
  - le sous-dossier *WEB-INF* ;
  - le fichier *index.jsp* servant à rediriger les utilisateurs vers la page d'accueil de l'application.

- **AgileReferential/src/main/webapp/media**

  Ce dossier contient l'ensemble des fichiers délivrés de manière statique par l'application web aux clients.

- **AgileReferential/src/main/webapp/media/css**

  Ce dossier contient l'ensemble des feuilles de style locales utilisées par l'application web.

- **AgileReferential/src/main/webapp/media/js**

  Ce dossier contient l'ensemble des fichiers JavaScript locaux utilisés par l'application web.

- **AgileReferential/src/main/webapp/WEB-INF**

  Ce dossier correspond à la racine de la partie privée de l'application web. Celui-ci contient :

  - le sous-dossier *views* ;
  - le fichier de configuration de Spring Security *app-agilia-security.xml* ;
  - le fichier de configuration de Spring MVC spring *app-agilia-servlet.xml* ;
  - le fichier de propriétés *config.properties* ;
  - le fichier descripteur de déploiement *web.xml*.

- **AgileReferential/src/main/webapp/WEB-INF/views**

  Ce dossier contient l'ensemble des vues de l'application web au format jsp.

- **AgileReferential/target**

  Ce dossier correspond au répertoire de travail de Maven. Celui-ci contient l'ensemble des éléments générés lors de la compilation et du déploiement de l'application.

3.4.10 La structure générale des pages

La structure générale des pages du prototype est similaire pour les ordinateurs, tablettes et smartphones. Elle décrit les zones principales qui architecturent les écrans de l'application.

Parmi ces zones, on trouve tout d'abord une aire de navigation en haut de page. Celle-ci est statique sur les ordinateurs et tablettes. Sur les smartphones, celle-ci est présentée sous la forme d'un menu déroulant. Elle contient une icône « Home » permettant de revenir à la page d'accueil à tout moment, ainsi que des liens vers les rubriques principales de l'application.

En dessous de l'espace de navigation, un fil d'Ariane indique à l'utilisateur ou il se situe dans l'arborescence du site. Il permet également à l'utilisateur de naviguer rapidement vers les niveau supérieurs.

Ensuite, une zone permettant d'afficher des messages d'alertes (succès, erreurs, informations) est prévue. Elle est optionnelle et uniquement présente à l'écran lorsqu'un message doit être montré à l'utilisateur.

Pour finir, le corps de page intègre tout le contenu et varie d'une page à l'autre.

Figure 20 : Structure générale des pages sur ordinateurs et tablettes

Figure 21 : Structure générale des pages sur Smartphones

4 Présentation du prototype

Ce chapitre a pour objectif de présenter le prototype réalisé.

Les fonctionnalités principales du prototype sont présentées au sein de ce chapitre. Notons que la présentation est générale et ne décrit pas en détails toutes les interactions et alternatives possibles.

Afin d'offrir un aperçu de l'application réalisée, nous avons sélectionné quelques écrans clés de cette dernière. Les visuels présentés correspondent à la version définie pour les ordinateurs et les tablettes. Une vue plus exhaustive des interfaces graphiques, incluant la version pour les smartphones est disponible dans les annexes (voir Annexe B).

4.1 Accès au référentiel

Lors de l'accès au référentiel, un utilisateur est dirigé vers la page d'accueil (Figure 22). Celle-ci affiche le nom du site ainsi que le slogan de manière concise et facilement lisible afin de donner une vision globale et claire des informations qu'il est possible de trouver.

Figure 22 : Prototype - Page d'accueil

4.2 Inscription

Un utilisateur anonyme peut, à partir du lien « S'inscrire » situé dans la barre de navigation, accéder à la page d'inscription afin de créer un compte qui lui permettra par la suite de s'authentifier et bénéficier de certains privilèges additionnels en fonction du rôle qui lui est octroyé (Figure 23). Par défaut, le rôle « Visiteur authentifié » est attribué aux utilisateurs lors de leur inscription. Tout autre rôle pourra être conféré ultérieurement par l'administrateur.

L'inscription consiste en l'encodage d'une adresse email, d'un mot de passe, du nom et du prénom de l'utilisateur. Tous ces champs sont obligatoires. Le mot de passe doit

contenir un minimum de six caractères et l'adresse email doit être valide et unique dans le système. Cette dernière servira d'identifiant. Dans le cas où une de ces règles n'est pas respectée, un message d'erreur général est affiché à l'écran ainsi qu'un message plus précis en dessous du/des champ(s) concerné(s) (Figure 24). Dans le cas où les données entrées sont valides, le compte est créé et l'utilisateur est redirigé vers la page d'authentification où il est notifié du succès de l'opération.

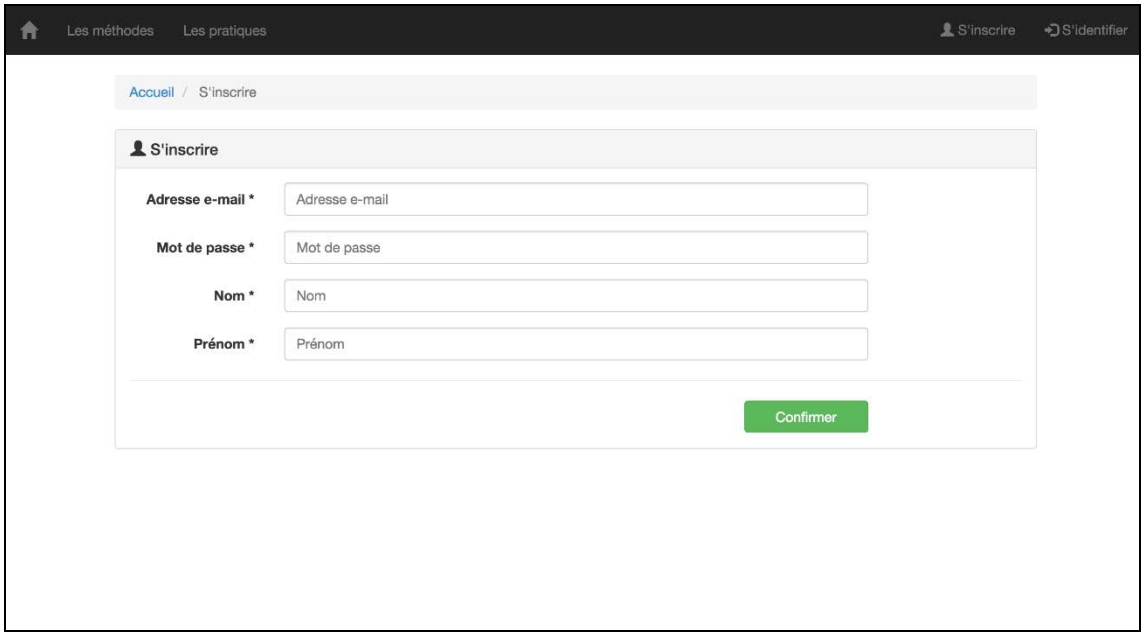


Figure 23 : Prototype - Page d'inscription

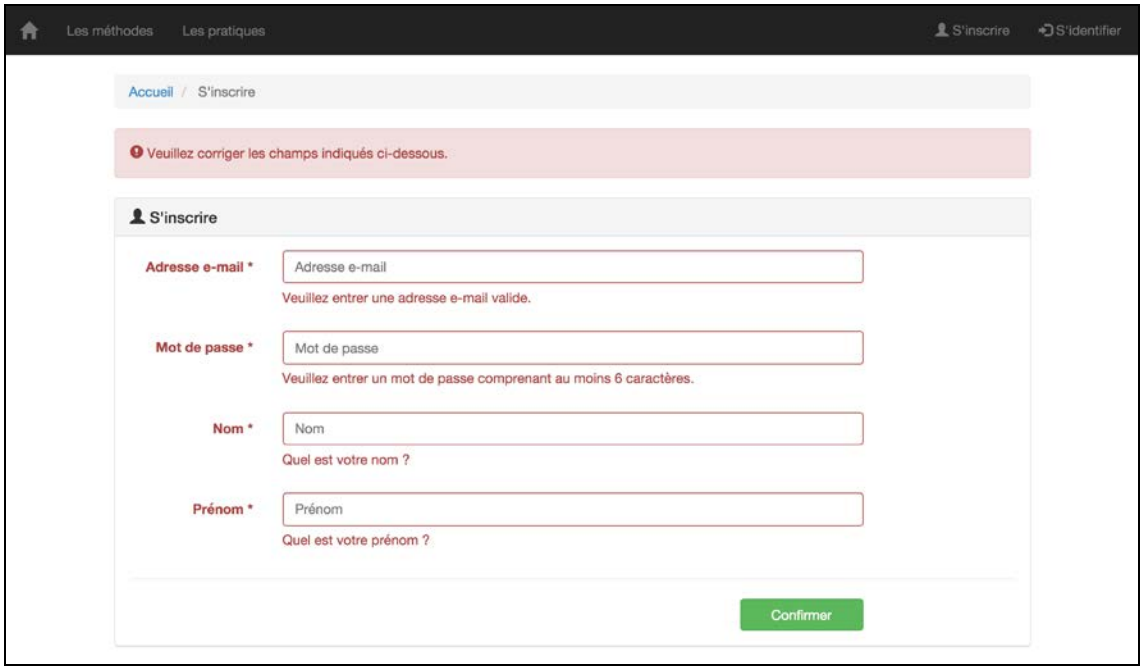


Figure 24 : Prototype - Page d'inscription - Erreurs dans le formulaire

4.3 Authentification

Une fois inscrit, un utilisateur anonyme peut tenter de s'authentifier. Pour ce faire, il doit introduire dans le formulaire d'authentification l'adresse email et le mot de passe qu'il a défini préalablement lors de son inscription (Figure 25). Si ceux-ci sont incorrects, un message d'erreur est affiché. Dans le cas contraire, il est authentifié et redirigé vers la page d'accueil où il est informé du succès de l'opération.

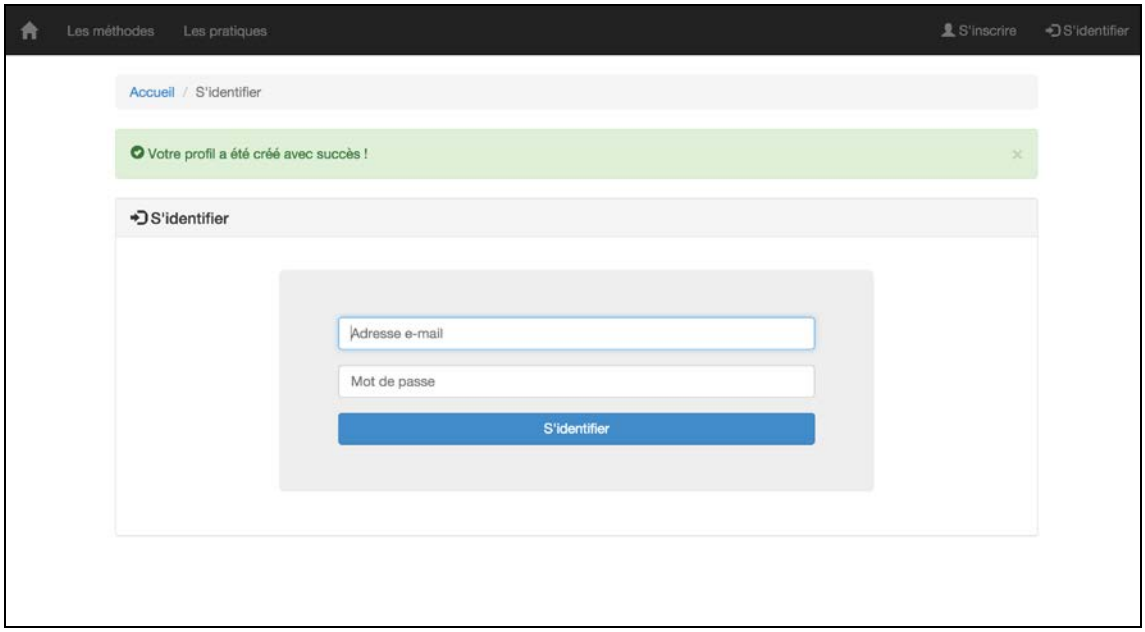


Figure 25 : Prototype - Page d'authentification

4.4 Aperçu des fiches du référentiel

Un utilisateur peut, à partir de la barre de navigation, accéder à l'aperçu des fiches du référentiel de type méthode (Figure 26) ou de type pratique (Figure 27).

L'aperçu est présenté sous forme d'une liste triée par ordre alphabétique. Un champ de recherche permet de filtrer cette dernière sur base d'un texte libre. Dans le cas où le référentiel ne contient aucune fiche du type concerné, un message d'information est affiché à l'écran et le champ de recherche est masqué.

Depuis cette écran, un utilisateur peut accéder aux informations détaillées d'une fiche ou si celui-ci est autorisé, accéder à la création d'une nouvelle fiche de ce même type. La consultation de l'aperçu des méthodes, ainsi que des pratiques est accessible par tous les utilisateurs authentifiés ou non. Cependant, l'option « Ajouter » n'est rendue disponible qu'aux utilisateurs authentifiés disposant du rôle « Editeur » ou « Administrateur ».

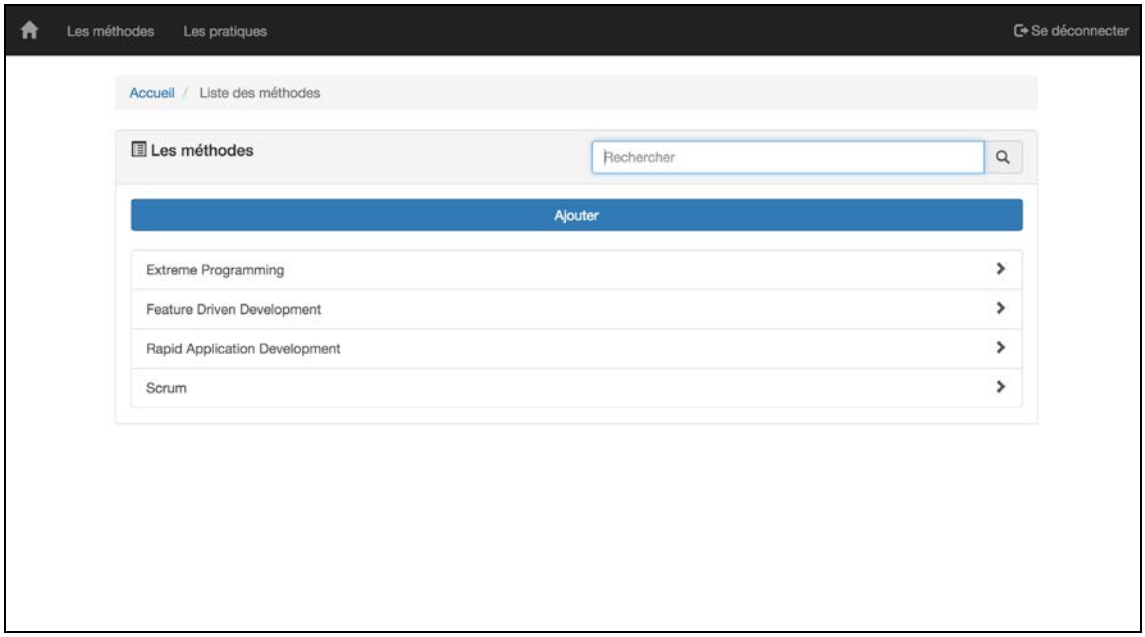


Figure 26 : Prototype - Page d'aperçu des fiches de type méthode

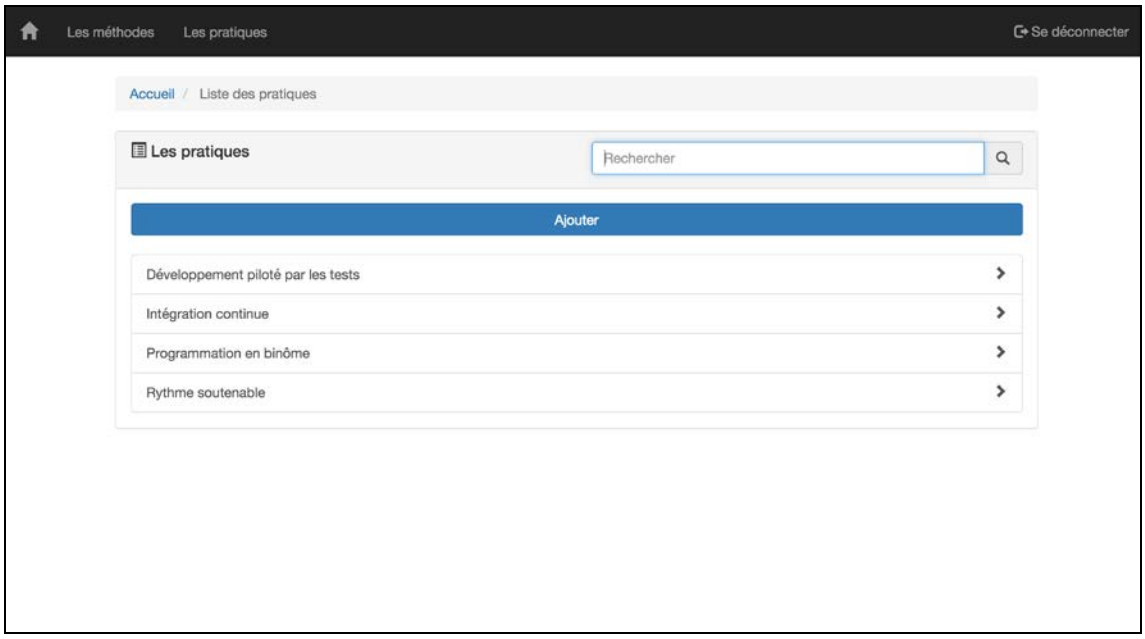


Figure 27 : Prototype - Page d'aperçu des fiches de type pratique

4.5 Consultation des détails d'une fiche du référentiel

En sélectionnant une fiche du référentiel au sein de l'aperçu, un utilisateur authentifié ou non peut accéder aux informations détaillées de cette dernière. Les informations sont groupées par section. Chacune d'entre elles est accompagnée d'une info-bulle indiquant le but de celle-ci (Figure 28 ; Figure 29).

Au sein de cette page, un utilisateur authentifié, quel que soit son rôle, peut introduire des commentaires à propos de la fiche courante. Toutefois, la possibilité d'éditer une

section ou de supprimer la fiche est limitée à un utilisateur authentifié doté du rôle « Editeur » ou « Administrateur ».

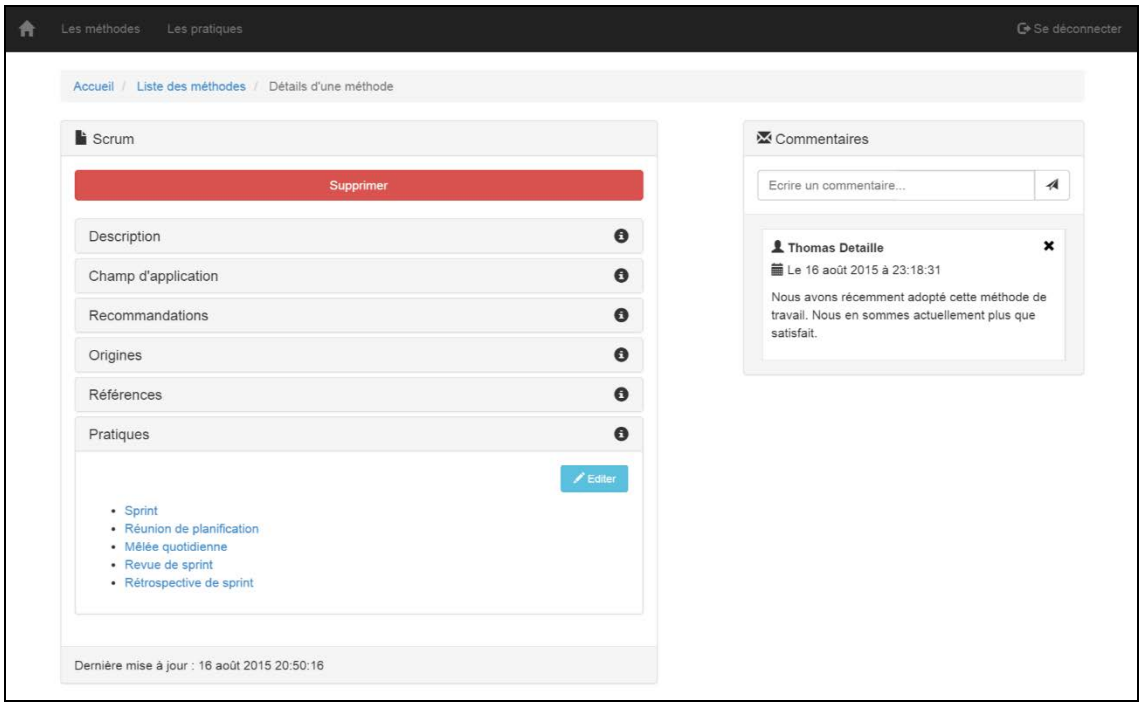


Figure 28 : Prototype - Page de consultation des détails d'une fiche de type méthode

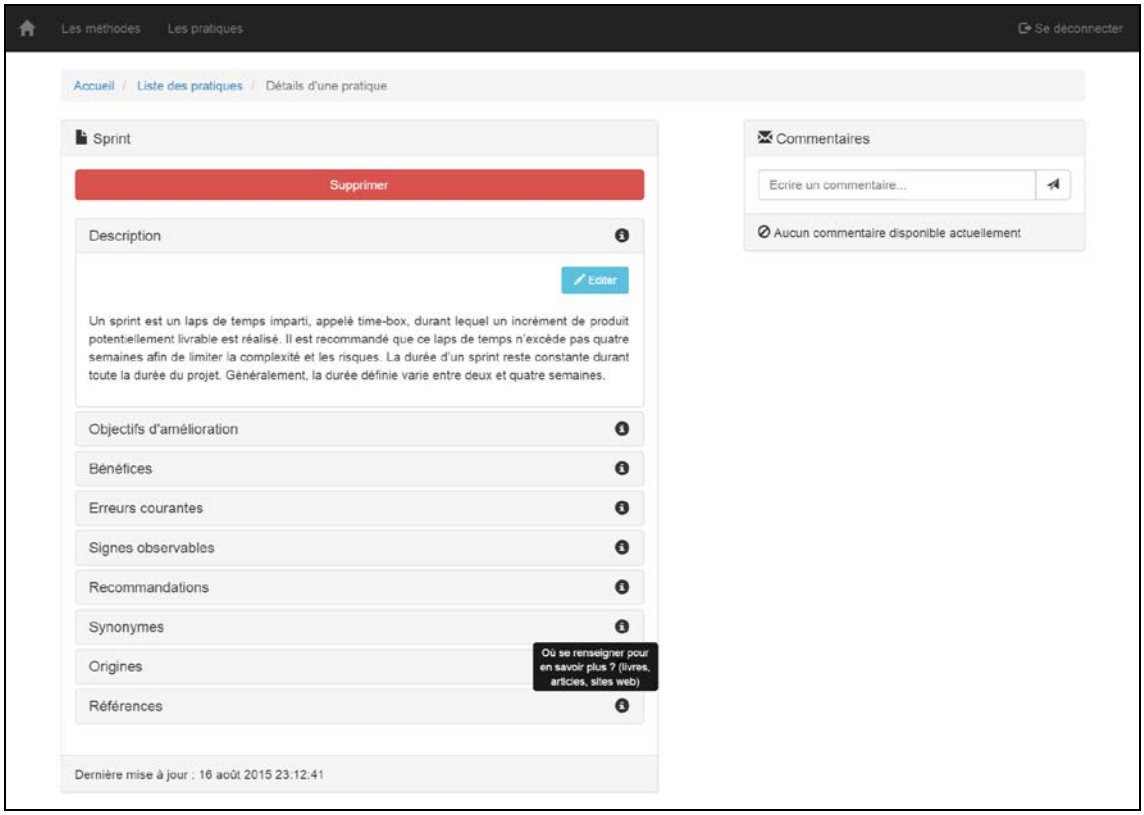


Figure 29 : Prototype - Page de consultation des détails d'une fiche de type pratique

4.6 Ajout d'une fiche dans le référentiel

Un utilisateur authentifié disposant du rôle « Editeur » ou « Administrateur » peut, à partir de l'écran « Aperçu », initier l'ajout de nouvelles fiches dans le référentiel. Deux types de fiches distinctes peuvent être créées : des fiches de type méthode (Figure 30) ou de type pratique (Figure 31).

La création d'une fiche se résume à l'introduction du nom de cette dernière. Celui-ci est obligatoire et doit être unique dans le système. Dans le cas où une de ces règles est enfreinte, une erreur est affichée. Sinon, la fiche est créée et l'utilisateur est dirigé vers les informations détaillées de celle-ci où il est convié à éditer ces dernières.

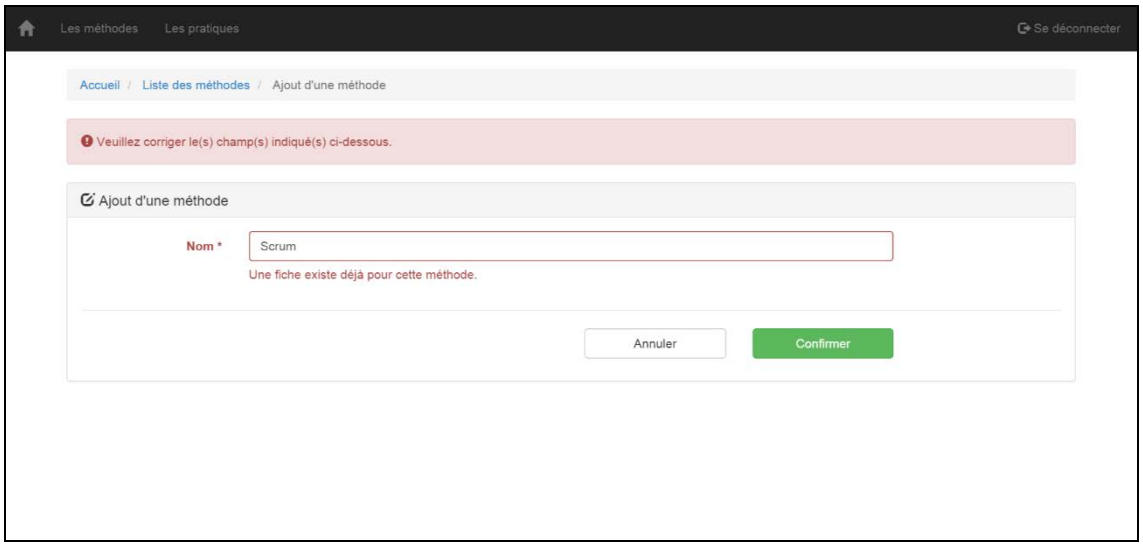


Figure 30 : Prototype - Page d'ajout d'une fiche de type méthode

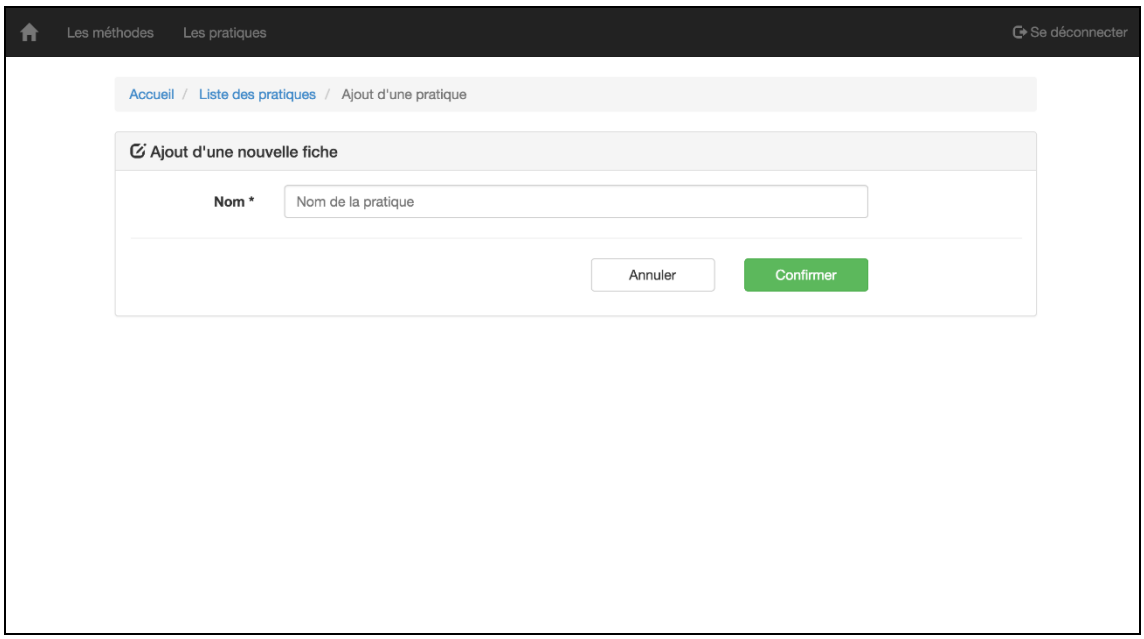


Figure 31 : Prototype - Page d'ajout d'une fiche de type pratique

4.7 Edition d'une fiche du référentiel

A partir des informations détaillées d'une fiche, un utilisateur authentifié disposant du rôle « Editeur » ou « Administrateur », peut procéder à l'édition de celle-ci section par section. Lorsqu'il clique sur le bouton « Editer », quelle que soit la section, une boîte de dialogue modale s'ouvre en mode plein écran et met à disposition un éditeur HTML de type WYSIWYG (What You See Is What You Get) (Figure 32). Grâce à cet outil, il est possible de rédiger, structurer et formater librement le contenu d'une section. Au cours de l'édition, un utilisateur peut soit annuler ou confirmer ses modifications. Dans les deux cas, la boite de dialogue modale se ferme et celui-ci revient sur la fiche concernée. Dans le premier cas, les données restent inchangées. Dans le second, les informations sont actualisées et un message indiquant le succès de l'opération est affiché.

Figure 32 : Prototype - Page d'édition d'une section d'une fiche

4.8 Suppression d'une fiche du référentiel

Un utilisateur authentifié disposant du rôle « Editeur » ou « Administrateur » peut initier la suppression d'une fiche du référentiel à partir de l'écran de consultation des informations détaillées de cette dernière.

Lorsque celui-ci indique qu'il souhaite procéder à la suppression d'une fiche du référentiel, une demande de confirmation est affichée sous la forme d'une boîte de dialogue modale (Figure 33). L'utilisateur peut soit annuler l'action, soit la confirmer. Dans le cas de la première option, la boîte de dialogue se ferme et celui-ci revient aux informations détaillées de la fiche. Dans l'alternative, la boîte de dialogue se clôt, la fiche est effacée et l'utilisateur est redirigé vers l'aperçu des fiches du type concerné où il sera notifié du succès de l'opération.

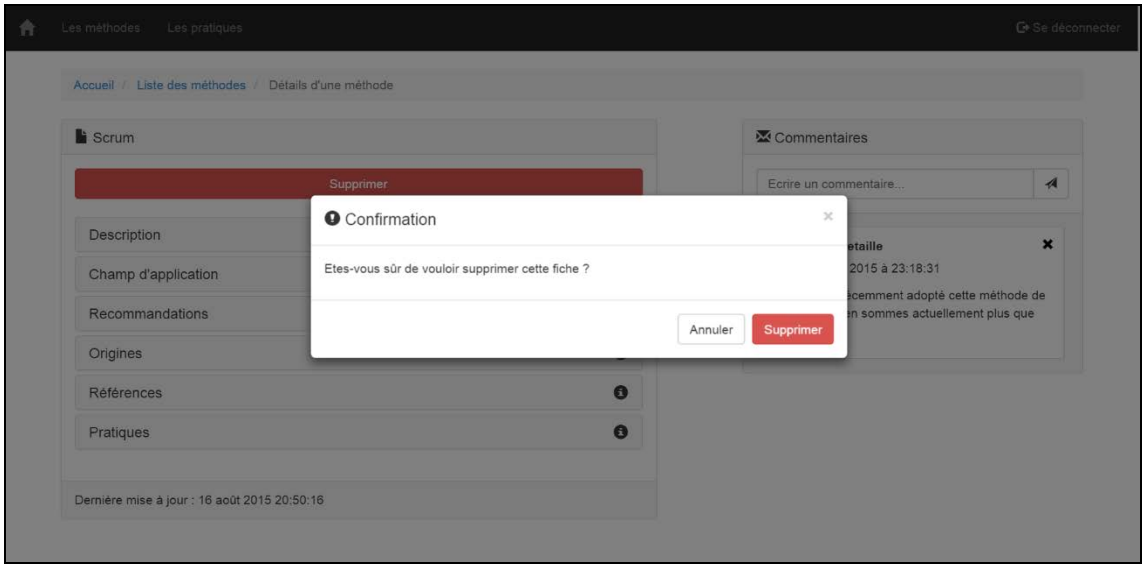


Figure 33 : Prototype - Page de confirmation de suppression d'une fiche

4.9 Association de fiches pratiques à une fiche méthode du référentiel

A partir des informations détaillées d'une fiche de type méthode, un utilisateur authentifié disposant du rôle « Editeur » ou « Administrateur », peut initier l'édition de la liste des pratiques qui composent la méthode. Lorsqu'il clique sur le bouton « Editer » au sein de la section « Pratiques », une boîte de dialogue modale s'ouvre en mode plein écran et met à disposition un outil permettant d'associer ou de dissocier des fiches de type pratique à la fiche de type méthode concernée (Figure 34).

Afin de faciliter la composition d'une méthode, trois composants sont mis à disposition de l'utilisateur :

1. Une liste déroulante permettant de filtrer la liste des pratiques. Celle-ci offre la possibilité de sélectionner soit toutes les pratiques, soit uniquement celles liées à une méthode particulière.

2. Un champ de recherche libre permettant de filtrer la liste des pratiques résultantes de la première sélection.

3. Une case à cocher permettant de sélectionner ou désélectionner la liste complète des pratiques affichées à l'écran et résultantes des filtrages opérés.

La combinaison de ces composants permet de procéder à la composition d'une méthode (hybride) de manière rapide et efficace.

Au cours de l'édition, un utilisateur peut soit annuler ou confirmer ses modifications. Dans les deux cas, la boite de dialogue modale se ferme et celui-ci revient sur la fiche concernée. Dans le premier cas, les données restent inchangées. Dans le second, les informations sont actualisées et un message indiquant le succès de l'opération est affiché.

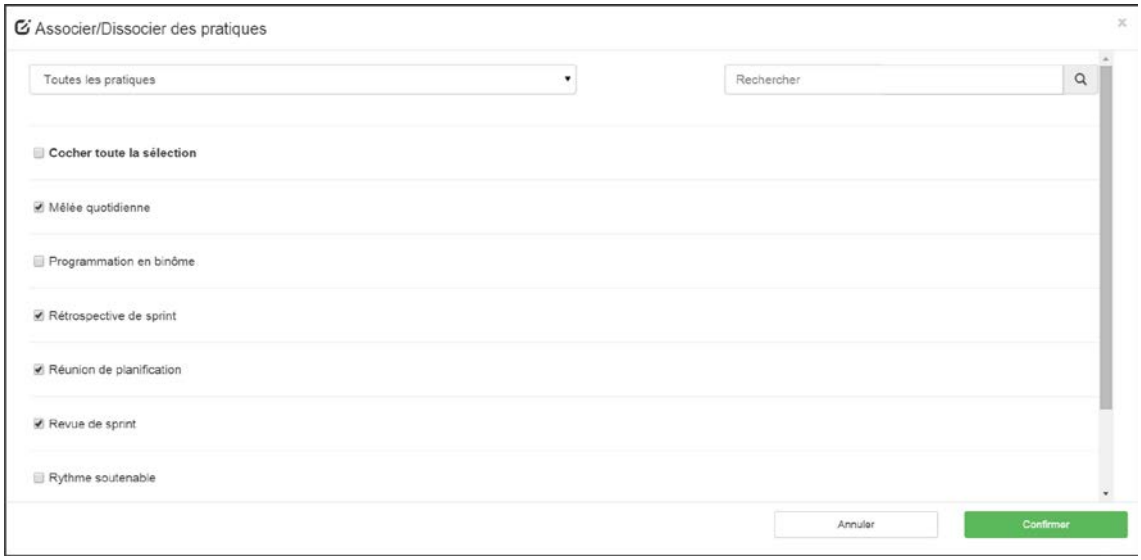


Figure 34 : Prototype - Page d'association des pratiques

4.10 Déconnexion

Un utilisateur authentifié peut à tout moment décider de se déconnecter. Pour ce faire, il lui suffit de cliquer sur le lien « Se déconnecter » situé dans la barre de navigation. Une fois cette action entreprise, la session de l'utilisateur est clôturée et celui-ci est redirigé vers la page d'authentification où il sera notifié du succès de l'opération.

5 Critique du prototype

Ce chapitre a pour objectif de mettre en évidence les points forts du prototype, les améliorations et extensions possibles, ainsi que les perspectives d'utilisation.

5.1 Points forts

5.1.1 Architecture

Toute application web développée avec un langage orienté-objet et qui se veut évolutive se doit d'être rigoureusement architecturée.

De ce fait, nous avons porté une attention particulière à la définition de l'architecture du prototype. Celle-ci est décrite en détails dans le chapitre 3 et correspond à une architecture 3-tiers supplémenté du patron de conception MVC (Modèle – Vue – Contrôleur) au niveau de la couche de présentation de cette dernière.

La mise en place d'une telle architecture logicielle rend l'application facilement extensible et maintenable.

5.1.2 Sécurité

Gestion des autorisations

La gestion des autorisations au sein d'une application est un concept important d'un point de vue sécuritaire. Elle consiste à vérifier que l'utilisateur a les permissions nécessaires pour accéder à une ressource ou effectuer une action précise.

Dans le cadre d'un référentiel, la qualité des données est essentielle. Dès lors, il est impératif de s'assurer que les données ne peuvent être ajoutées, modifiées et supprimées que par un groupe défini et identifiable d'utilisateurs.

Pour cela, nous avons mis en place un contrôle d'accès à base de rôles (Role Based Access Control).

Sur base des rôles définis et des droits associés à ceux-ci, nous avons tout d'abord personnalisé le contenu des pages afin de limiter l'accès à certaines fonctionnalités selon le rôle et les droits d'accès de l'utilisateur.

Ensuite, nous avons sécurisé l'accès aux pages web de l'application en effectuant un contrôle au niveau des URLs afin de protéger les pages contre des accès directs sans autorisation.

Finalement, nous avons procédé à la sécurisation des appels de méthodes, c'est-à-dire, renforcé la sécurité en effectuant un contrôle au niveau de la couche métier. Dès lors, seuls les utilisateurs authentifiés et disposant des droits nécessaires auront la possibilité d'exécuter une méthode sécurisée.

Chiffrement des mots de passe

Les mots de passe des utilisateurs étant des données sensibles, nous les avons chiffrés afin de les stocker de manière sécurisée dans la base de données.

Pour ce faire, nous avons utilisé l'algorithme de chiffrement « bcrypt ». Celui-ci est un algorithme de calcul d'empreinte lent. L'avantage de ce dernier est qu'il rend le procédé de brute-force beaucoup plus long qu'avec un algorithme tel que MD5.

5.1.3 Ergonomie

La conception des interfaces graphiques a été réalisée de façon à rendre l'utilisation agréable et triviale. Pour cela, nous avons :

- Réalisé des interfaces graphiques sobres, c'est-à-dire simples et peu chargées ;
- Respecté une homogénéité au niveau de la structure des pages ;
- Mis à disposition un menu de navigation permettant d'accéder aux principales rubriques du site sur toutes les pages de l'application ou encore de revenir à la page d'accueil de celle-ci à tout moment ;
- Mis en place un fil d'Ariane facilitant la navigation et situant les utilisateurs dans l'arborescence du site.

De plus, les interfaces graphiques ont été réalisées de manière à optimiser le rendu visuel sur les différents types d'appareils couramment utilisés tels que les ordinateurs de bureau, les ordinateurs portables, les tablettes et les smartphones.

Toutefois, sur base du retour des utilisateurs ayant testé le prototype[4], nous avons constaté que le mode édition sur les appareils mobiles peut nécessiter un certain temps d'adaptation pour les usagers néophytes.

Notons également que l'ergonomie peut encore être améliorée. Pour cela, l'idéal serait notamment d'effectuer plusieurs vagues successives de tests avec différents panels d'utilisateurs afin d'effectuer des améliorations par touches progressives sur base des retours collectés.

[4] Sept personnes dans notre entourage ont testé le prototype durant sa réalisation.

5.1.4 Portabilité

Le prototype réalisé étant une application web adaptative, il est compatible avec la plupart des navigateurs et appareils récents du marché.

Du côté des navigateurs, nous avons testé celui-ci sur la dernière version disponible des navigateurs suivants :

- Internet Explorer ;
- Mozilla Firefox ;
- Safari ;
- Chrome ;

Du côté des appareils, nous avons testé celui-ci sur divers appareils tels que :

- Ordinateur sous Windows ;
- Ordinateur sous Mac OS ;
- iPad mini sous iOS ;
- iPhone 5 / iPhone 6 sous iOS ;
- Samsung S4 mini sous Android ;

Notons que quelques bugs visuels mineurs sont encore à résoudre. Nous avons par exemple constaté un problème de rognure des textes lors de l'affichage des infobulles sur le Samsung S4 mini tournant sous Android.

5.1.5 Internationalisation (i18n)

Le prototype réalisé n'est actuellement disponible que dans la langue française. Cependant, celui-ci pourrait très bien être utilisé dans d'autres langues. C'est pourquoi, nous avons implémenté celui-ci de manière à ce qu'il soit aisé de modifier la langue.

Pour ce faire, nous avons basé l'affichage des messages sur des fichiers de propriétés contenant les traductions. De cette façon, le portage vers une autre langue se traduit par l'adaptation ou l'ajout d'un nouveau fichier de traductions ainsi que quelque peu de configuration.

5.2 Améliorations et extensions possibles

5.2.1 Extension du périmètre actuel du référentiel

Le périmètre actuel du prototype couvre les méthodes et pratiques agiles. Cependant, comme stipulé dans la vision du produit, nous estimons qu'il serait intéressant d'étendre le périmètre du référentiel avec d'autres concepts afférant aux méthodes agiles tels que les rôles et les artéfacts.

De la sorte, il serait possible de documenter les méthodes de façon plus complète en définissant l'ensemble des rôles officiels d'une méthode et les principales responsabilités qui leur sont liées ainsi que les différents artéfacts consommés et produits.

5.2.2 Support à l'adoption partielle des pratiques agiles

Dans le cadre du support de l'adoption des méthodes agiles et plus particulièrement de l'adoption partielle des pratiques agiles, il serait profitable d'ajouter une fonctionnalité permettant de proposer aux utilisateurs des schémas de stratégie d'adoption partielle des pratiques agiles en fonction d'un objectif d'amélioration choisi.

De manière plus concrète, l'utilisateur serait invité à sélectionner un objectif d'amélioration tel que la réduction du time to market parmi une liste prédéfinie. En fonction de l'objectif choisi, un schéma de stratégie d'adoption partielle comme défini par Amr Elssamadisy dans son ouvrage « Agile Adoption Patterns : A road to organizationnal success » serait proposé à l'utilisateur [Elssamadisy, 2009]. De plus, chaque pratique existante au sein du référentiel et reprise dans le schéma affiché serait mise en évidence et pourrait être consultée directement à partir de ce dernier.

5.2.3 Validation instantanée

La validation sur les champs des formulaires du prototype est actuellement réalisée au moment de la soumission de ceux-ci. Afin de maximiser l'expérience utilisateur, il serait judicieux d'implémenter un système de validation en temps réel.

Pour ce faire, nous pourrions entre autres recourir à la méthode de développement web AJAX (Asynchronous Javascript and XML).

Ainsi nous pourrions, par exemple, vérifier l'existence ou non d'un compte utilisateur en temps réel lors de l'inscription ou encore l'existence d'un nom de pratique ou méthode lors de la création d'une nouvelle fiche au sein du référentiel.

5.2.4 Ajout d'un module de gestion des utilisateurs

Dans le cadre d'une exploitation réelle du référentiel, il s'avèrerait impératif d'implémenter un module de gestion des utilisateurs. Ce dernier serait, bien entendu, accessible uniquement par l'administrateur et permettrait de gérer les comptes et les rôles des utilisateurs.

5.2.5 Ajout d'un module de retour d'expérience

Le référentiel fournit un cadre de référence théorique. Cependant, il peut être notamment intéressant de disposer d'informations pratiques.

Pour cela, un module permettant aux utilisateurs de fournir et consulter des retours d'expérience quant à l'adoption des pratiques et/ou des méthodes agiles pourrait apporter une plus-value et s'avérer complémentaire au référentiel.

5.3 Perspectives d'utilisation

5.3.1 Référentiel commun public

Une première perspective d'utilisation est une utilisation en tant que référentiel commun public, c'est-à-dire, un référentiel à propos des méthodes et pratiques agiles officielles existantes mis en ligne et accessible en consultation par tout un chacun.

5.3.2 Référentiel commun d'entreprise

Une seconde perspective d'utilisation est une utilisation en tant que référentiel commun d'entreprise, c'est-à-dire, comme outil permettant à une organisation de documenter en détails la/les méthodologie(s) de travail employée(s) au sein de celle-ci. Une telle exploitation de l'outil servirait d'une part à formaliser les méthodes et pratiques à appliquer au sein de la firme, mais également d'en communiquer les détails aux différents membres de l'organisation sous la forme d'un guide méthodologique.

5.3.3 Outil d'aide à la composition d'une méthode

Une troisième et dernière perspective d'utilisation est une utilisation en tant qu'outil d'aide à la composition d'une méthode dans la même lignée que le composeur EPF[5] (Eclipse Process Framework). Dans cette optique, le référentiel contiendrait l'ensemble des méthodes et pratiques agiles existantes. Ensuite, ces données serviraient de base pour faciliter la composition et la description de méthodes personnalisées.

[5] http://www.eclipse.org/epf/

6 Conclusion

L'objectif principal de ce travail était de dresser un état des lieux général des méthodes agiles et de l'adoption de ces dernières, ainsi que de proposer, concevoir et réaliser un prototype de référentiel web des méthodes et pratiques agiles.

Au cours de ce travail, lors de la première phase de revue de la littérature, nous avons constaté que l'adoption des méthodes agiles était actuellement en plein essor suite, entre autres, aux enjeux grandissants engendrés par la digitalisation et la « nouvelle économie ».

Nous avons également remarqué qu'il existe de nombreuses méthodes agiles. Chacune d'entre elles possède ses spécificités. Néanmoins, toutes respectent le paradigme agile défini au sein du manifeste agile à travers quatre valeurs et douze principes sous-jacents.

De plus, nous avons découvert que la transition vers l'agilité n'est pas un processus défini générique et universel. En effet, une organisation peut choisir d'adopter une méthode particulière, une combinaison de méthodes ou une collection de pratiques, consciencieusement sélectionnées, en fonction de son contexte organisationnel et de ses objectifs d'amélioration.

En ce qui concerne le prototype réalisé, celui-ci est fonctionnel et peut être utilisé comme référentiel et/ou composeur de méthodes. Il constitue notamment un noyau solide pour la mise en place d'un projet de plus grande envergure. Nous proposons d'ailleurs diverses améliorations et extensions possibles telles que l'ajout de concepts supplémentaires ou encore l'ajout de nouveaux modules permettant de gérer les comptes utilisateurs ou de collecter des retours d'expérience quant à l'adoption et l'utilisation des méthodes et pratiques agiles. Les évolutions futures seront facilitées par l'architecture souple et évolutive que nous avons mise en place.

D'un point de vue personnel, la réalisation de ce travail nous a permis d'approfondir nos connaissances sur le thème de l'agilité, de combiner nos compétences fonctionnelles et techniques, d'apprendre de nouvelles technologies, mais également d'appliquer différents concepts et techniques abordés dans le cursus que nous avons suivi.

Pour conclure, le prototype mis en place ouvre la porte à des perspectives diverses et variées. Une des principales perspectives étant de faciliter les processus liés à l'adoption des méthodes agiles en offrant non seulement un cadre de référence commun, mais également un outil d'aide à la décision et composition d'une méthode agile adaptée au contexte organisationnel.

Références bibliographiques

[Abrahamsson et al, 2002] Abrahamsson P., Salo O., Ronkainen J., Warsta J., *Agile software development methods*, VTT, Espoo, Finland, 2002.

[Académie Française, 2015] Académie Française, Dictionnaire de l'académie française : neuvième édition (Version informatisée), http://atilf.atilf.fr/academie9.htm, (Consulté en mars 2015).

[Agile Alliance, 2015] Agile Alliance, Guide to Agile Practices, http://guide.agilealliance.org, 2015 (Consulté en mars 2015).

[Beck, 2000] Beck K., *Extreme programming explained : Embrace change*, Addison-Wesley, Reading, Massachusset, 2000.

[Beck et al, 2001] Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R., Mellor S., Schwaber K., Sutherland J., Thomas D., Manifesto for Agile Software Development, http://www.agilemanifesto.org/iso/fr/, 2001 (Consulté en janvier 2015).

[Beck et al, 2005] Beck, K. et Andres C., *Extreme programming explained : Embrace change (Second edition)*, Addison-Wesley, Boston, Massachusset, 2005.

[Boisvert et al, 2011] Boisvert M., Trudel S., *Choisir l'agilité*, Dunod, Paris, 2011.

[Cockburn, 2005] Cockburn A., *Crystal clear : a human-powered methodology for small teams*, Addison-Wesley, Boston, 2005.

[Coq, 2012] Coq T., *Méthodes et informatique*, Hermès science publications, Paris, 2012.

[Daniel et al, 2013] Daniel F., Dolog P., Li Q., *Web Engineering: 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013, Proceedings*, Springer, Berlin, 2013.

[Elssamadisy, 2009] Elssamadisy A., *Agile adoption patterns*, Addison-Wesley, Upper Saddle River, New Jersey, 2009.

[Epf.eclipse.org, 2008] Epf.eclipse.org, Eclipse Process Framework Project (EPF), http://www.eclipse.org/epf/, 2008 (Consulté en août 2015).

[Epf.eclipse.org, 2012] Epf.eclipse.org, OpenUP, http://epf.eclipse.org/wikis/openup, 2012 (Consulté en mars 2015).

[Granddictionnaire, 2015] Granddictionnaire, Le grand dictionnaire terminologique, http://www.granddictionnaire.com/index.aspx, 2015 (Consulté en juin 2015).

[Highsmith, 2010] Highsmith J., *Adaptive software development*, Dorset House Pub, New York, 2000.

[IBM Corporation, 2007] IBM Corporation, XP Artifacts, http://epf.eclipse.org/wikis/xp/, 2007 (Consulté en février 2015).

[Ijab et al, 2004] Ijab M., Hamid S., « Basic artifacts of extreme programming », *New Straits Time*, page 26, Juin 2004.

[IT Knowledge Portal, 2015] IT Knowledge Portal, IT Standards and Methodologies, http://www.itinfo.am/eng/software-development-methodologies/, 2015 (Consulté en août 2015).

[Jeffries, 2015] Jeffries R., What is Extreme Programming?, http://ronjeffries.com/xprog/what-is-extreme-programming/, 2011 (Consulté en février 2015).

[Larousse, 2015] Larousse, Dictionnaire de français, http://www.larousse.fr/, (Consulté en mai 2015).

[Messager Rota, 2009] Messager Rota V., *Gestion de projet : Vers les méthodes agiles*, Eyrolles, Paris, 2009.

[Palmer et al, 2002] Palmer S., Felsing J., *A practical guide to Feature-driven Development*, Prentice Hall, Upper Saddle River, New Jersey, 2002

[Schwaber, 2004] Schwaber K., *Agile project management with Scrum*, Microsoft Press, Redmond, Washington, 2004.

[Schwaber, 2009] Schwaber K., S*crum Guide*, Scrum Alliance, 2009.

[Schwaber et al, 2013] Schwaber K., Sutherland J., The Scrum Guide, http://www.scrumguides.org/scrum-guide.html, 2013, (Consulté en février 2015).

[Stapleton, 1997] Stapleton J., *DSDM (Dynamic Systems Development Method)*, Addison-Wesley, Harlow, United Kingdom, 1997.

[Vickoff, 2009] Vickoff J., *Méthode Agile, Les meilleures pratiques, Compréhension et mise en œuvre*, Agile Alliance, 2009.

[Vickoff, 2014] Vickoff J., AGILE Historique et évolution, http://www.entreprise-agile.com/HistoAgile.pdf, 2014, (Consulté en février 2015).

[Wells, 2013] Wells D., Extreme Programming : A Gentle Introduction, http://www.extremeprogramming.org/, 2013 (Consulté en février 2015).

[West et al, 2010] West D., Grant T., Gerush M., D'Silva D., *Agile development : Mainstream adoption has changed agility*, Forrester Research, 2010.

Annexes

A. Code source du prototype réalisé
B. Interfaces graphiques du prototype réalisé

Université de Namur
Faculté d'informatique
Année académique 2014 – 2015

Mise en place d'un référentiel Web des méthodes et pratiques agiles

ANNEXES

Thomas Detaille

Promoteur : Naji Habra

Co-promoteur : Hajer Ayed

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatique

ANNEXE A

Code source du prototype réalisé

Table des matières

1 Entités

1.1 User.java

```
package com.unamur.entity;

import java.sql.Timestamp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

import org.hibernate.validator.constraints.NotEmpty;

/**
 * User entity - Represents a user
 * */
@Entity
@ Table(name = "user")
public class User {

  @Id
  @ GeneratedValue
  @ Column(name = "id")
  private Integer id;

  @Column(name = "email", unique = true)
  @ Pattern(regexp =
    "^[\\w-]+(\\.[\\w-]+)*@([a-z0-9-]+(\\.[a-z0-9-]+)*?\\.[a-z]{2,6}|(\\d{1,3}\\.){3}\\d{1,3})(:\\d{4})?$")
  @ Size(max = 128)
  private String email;

  @Column(name = "first_name")
  @ NotEmpty
  @ Size(max = 64)
  private String firstName;

  @Column(name = "last_name")
  @ NotEmpty
  @ Size(max = 64)
  private String lastName;

  @Column(name = "password")
  @ Size(min = 6, max = 64)
  private String password;

  @ManyToOne
  @ JoinColumn(name = "role_id", referencedColumnName = "id")
  private Role role;
```

```
@Column(name = "creation_date")
private Timestamp creationDate;

@Column(name = "last_modification_date")
private Timestamp lastModificationDate;

public Integer getId() {
  return id;
}

public void setId(Integer id) {
  this.id = id;
}

public String getEmail() {
  return email;
}

public void setEmail(String email) {
  this.email = email;
}

public String getFirstName() {
  return firstName;
}

public void setFirstName(String firstName) {
  this.firstName = firstName;
}

public String getLastName() {
  return lastName;
}

public void setLastName(String lastName) {
  this.lastName = lastName;
}

public String getPassword() {
  return password;
}

public void setPassword(String password) {
  this.password = password;
}

public Role getRole() {
  return role;
}

public void setRole(Role role) {
  this.role = role;
}

public Timestamp getCreationDate() {
  return creationDate;
}
```

```
  public void setCreationDate(Timestamp creationDate) {
    this.creationDate = creationDate;
  }

  public Timestamp getLastModificationDate() {
    return lastModificationDate;
  }

  public void setLastModificationDate(Timestamp lastModificationDate) {
    this.lastModificationDate = lastModificationDate;
  }
}
```

1.2 Role.java

```
package com.unamur.entity;

import java.sql.Timestamp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

/**
 * Role entity - Represents a user role
 * */
@Entity
@Table(name = "role")
public class Role {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;

    @Column(name = "role")
    private String role;

    @Column(name = "creation_date")
    private Timestamp creationDate;

    @Column(name = "last_modification_date")
    private Timestamp lastModificationDate;

    public Integer getId() {
            return id;
    }

    public void setId(Integer id) {
            this.id = id;
    }

    public String getRole() {
            return role;
    }
```

```
    public void setRole(String role) {
        this.role = role;
    }

    public Timestamp getCreationDate() {
        return creationDate;
    }

    public void setCreationDate(Timestamp creationDate) {
        this.creationDate = creationDate;
    }

    public Timestamp getLastModificationDate() {
        return lastModificationDate;
    }

    public void setLastModificationDate(Timestamp lastModificationDate) {
        this.lastModificationDate = lastModificationDate;
    }
}
```

1.3 Method.java

```
package com.unamur.entity;

import java.sql.Timestamp;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.OneToMany;
import javax.persistence.OrderBy;
import javax.persistence.Table;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

/**
 * Method entity - Represents a Method.
 * */
@Entity
@Table(name = "method")
public class Method {

    @Id
    @GeneratedValue
    @Column(name = "method_id")
    private Integer id;

    @Column(name = "name")
```

```
@NotEmpty
@Length(max = 40)
private String name;

@Column(name = "description")
private String description;

@Column(name = "scope")
private String scope;

@Column(name="historical_context")
private String historicalContext;

@Column(name="adoption_recommendations")
private String adoptionRecommendations;

@Column(name="bibliography")
private String bibliography;

@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(name="method_practice",
        joinColumns={@JoinColumn(name="method_id")},
        inverseJoinColumns={@JoinColumn(name="practice_id")})
private Set<Practice> practices = new HashSet<Practice>();

@OneToMany(fetch = FetchType.EAGER, mappedBy="method")
@OrderBy("post_on DESC")
private Set<MethodComment> comments;

@Column(name = "creation_date")
private Timestamp creationDate;

@Column(name = "last_modification_date")
private Timestamp lastModificationDate;

public Timestamp getCreationDate() {
   return creationDate;
}

public void setCreationDate(Timestamp creationDate) {
   this.creationDate = creationDate;
}

public Timestamp getLastModificationDate() {
   return lastModificationDate;
}

public void setLastModificationDate(Timestamp lastModificationDate) {
   this.lastModificationDate = lastModificationDate;
}

public String getName() {
        return name;
}

public Integer getId() {
        return id;
```

```
}

public void setId(Integer id) {
        this.id = id;
}

public String getScope() {
   return scope;
}

public void setScope(String scope) {
   this.scope = scope;
}

public String getHistoricalContext() {
   return historicalContext;
}

public void setHistoricalContext(String historicalContext) {
   this.historicalContext = historicalContext;
}

public String getAdoptionRecommendations() {
   return adoptionRecommendations;
}

public void setAdoptionRecommendations(String adoptionRecommendations) {
   this.adoptionRecommendations = adoptionRecommendations;
}

public String getBibliography() {
   return bibliography;
}

public void setBibliography(String bibliography) {
   this.bibliography = bibliography;
}

public void setName(String name) {
        this.name = name;
}

public String getDescription() {
        return description;
}

public void setDescription(String description) {
        this.description = description;
}

public Set<Practice> getPractices() {
   return practices;
}

public void setPractices(Set<Practice> practices) {
   this.practices = practices;
}
```

```
    public Set<MethodComment> getComments() {
        return comments;
    }

    public void setComments(Set<MethodComment> comments) {
        this.comments = comments;
    }
}
```

1.4 Practice.java

```
package com.unamur.entity;

import java.sql.Timestamp;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.OneToMany;
import javax.persistence.OrderBy;
import javax.persistence.Table;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

/**
 * Practice entity - Represents a Practice.
 * */
@Entity
@Table(name = "Practice")
public class Practice {

    @Id
    @GeneratedValue
    @Column(name = "practice_id")
    private Integer id;

    @Column(name = "name")
    @NotEmpty
    @Length(max = 40)
    private String name;

    @Column(name = "description")
    private String description;

    @Column(name = "benefits")
    private String benefits;

    @Column(name = "synonyms")
    private String synonyms;

    @Column(name = "bibliography")
    private String bibliography;
```

```
@Column(name = "improvement_targets")
private String improvementTargets;

@Column(name = "adoption_recommendations")
private String adoptionRecommendations;

@Column(name = "observable_signs")
private String observableSigns;

@Column(name = "historical_context")
private String historicalContext;

@Column(name = "common_mistakes")
private String commonMistakes;

@ManyToMany(fetch = FetchType.EAGER, mappedBy = "practices")
private Set<Method> methods = new HashSet<Method>();

@OneToMany(fetch = FetchType.EAGER, mappedBy="practice")
@OrderBy("post_on DESC")
private Set<PracticeComment> comments;

@Column(name = "creation_date")
private Timestamp creationDate;

@Column(name = "last_modification_date")
private Timestamp lastModificationDate;

public Timestamp getCreationDate() {
   return creationDate;
}

public void setCreationDate(Timestamp creationDate) {
   this.creationDate = creationDate;
}

public Timestamp getLastModificationDate() {
   return lastModificationDate;
}

public void setLastModificationDate(Timestamp lastModificationDate) {
   this.lastModificationDate = lastModificationDate;
}

public String getImprovementTargets() {
        return improvementTargets;
}

public void setImprovementTargets(String improvementTargets) {
        this.improvementTargets = improvementTargets;
}

public String getAdoptionRecommendations() {
        return adoptionRecommendations;
}
```

```
public void setAdoptionRecommendations(
            String adoptionRecommendations) {
        this.adoptionRecommendations = adoptionRecommendations;
}

public String getObservableSigns() {
        return observableSigns;
}

public void setObservableSigns(String observableSigns) {
        this.observableSigns = observableSigns;
}

public String getHistoricalContext() {
        return historicalContext;
}

public void setHistoricalContext(String historicalContext) {
        this.historicalContext = historicalContext;
}

public String getBibliography() {
        return bibliography;
}

public void setBibliography(String bibliography) {
        this.bibliography = bibliography;
}

public String getCommonMistakes() {
        return commonMistakes;
}

public void setCommonMistakes(String commonMistakes) {
        this.commonMistakes = commonMistakes;
}

public Integer getId() {
        return id;
}

public void setId(Integer id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getDescription() {
        return description;
}
```

```
public void setDescription(String description) {
        this.description = description;
}

public String getBenefits() {
        return benefits;
}

public void setBenefits(String benefits) {
        this.benefits = benefits;
}

public Set<Method> getMethods() {
        return methods;
}

public void setMethods(Set<Method> methods) {
        this.methods = methods;
}

public String getSynonyms() {
        return synonyms;
}

public void setSynonyms(String synonyms) {
        this.synonyms = synonyms;
}

public Set<PracticeComment> getComments() {
   return comments;
}

public void setComments(Set<PracticeComment> comments) {
   this.comments = comments;
}

@Override
public int hashCode() {

        if (id == null)
           return 0;
        else
           return new Integer(id).hashCode();
}

@Override
public boolean equals(Object obj) {

        if (obj == null) {
           return false;
        }

        if (!(obj instanceof Practice)) {
           return false;
        }

        return this.id == ((Practice) obj).getId();
```

```
    }
}
```

1.5 MethodComment.java

```
package com.unamur.entity;

import java.sql.Timestamp;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.validator.constraints.Length;

/**
 * MethodComment entity - Represents a comment made on a method.
 * */
@Entity
@Table(name = "comment_on_method")
public class MethodComment {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;

    @Column(name = "message")
    @Length(max = 255)
    private String message;

    @ManyToOne
    @JoinColumn(name = "author", referencedColumnName = "id")
    private User author;

    @Column(name = "post_on")
    private Timestamp postDate;

    @ManyToOne
    @JoinColumn(name = "method_id", referencedColumnName = "method_id")
    private Method method;

    public Integer getId() {
        return id;
    }

    public Method getMethod() {
        return method;
    }

    public void setMethod(Method method) {
        this.method = method;
    }

    public void setId(Integer id) {
```

```
        this.id = id;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public User getAuthor() {
        return author;
    }

    public void setAuthor(User author) {
        this.author = author;
    }

    public void setPostDate(Timestamp postDate) {
        this.postDate = postDate;
    }

    public Timestamp getPostDate() {
        return postDate;
    }
}
```

1.6 PracticeComment.java

```
package com.unamur.entity;

import java.sql.Timestamp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.validator.constraints.Length;

/**
 * PracticeComment entity - Represents a comment made on a practice.
 * */
@Entity
@Table(name = "comment_on_practice")
public class PracticeComment {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;

    @Column(name = "message")
    @Length(max = 255)
```

```
    private String message;

    @ManyToOne
    @JoinColumn(name = "author", referencedColumnName = "id")
    private User author;

    @Column(name = "post_on")
    private Timestamp postDate;

    @ManyToOne
    @JoinColumn(name = "practice_id", referencedColumnName = "practice_id")
    private Practice practice;

    public Integer getId() {
            return id;
    }

    public Practice getPractice() {
            return practice;
    }

    public void setPractice(Practice practice) {
            this.practice = practice;
    }

    public void setId(Integer id) {
            this.id = id;
    }

    public String getMessage() {
            return message;
    }

    public void setMessage(String message) {
            this.message = message;
    }

    public User getAuthor() {
            return author;
    }

    public void setAuthor(User author) {
            this.author = author;
    }

    public void setPostDate(Timestamp postDate) {
            this.postDate = postDate;
    }

    public Timestamp getPostDate() {
            return postDate;
    }
}
```

2 DAO - Interfaces

2.1 UserDao.java

```
package com.unamur.dao;

import com.unamur.entity.User;

public interface UserDao {

   /**
    * This method inserts a new user in the database.
    *
    * @param user - The user to insert.
    */
   public void createUser(User user);

   /**
    * This method fetches a user in the database based on his email address.
    *
    * @param email - The email of the user to fetch.
    * @return A User object if found. Else null.
    */
   User getUserByEmail(String email);

}
```

2.2 MethodDao.java

```
package com.unamur.dao;

import java.util.List;

import com.unamur.entity.Method;

public interface MethodDao {

   /**
    * This method creates a method in the database.
    *
    * @param method
    *          - The method to create.
    * @return The identifier of the newly created method.
    */
   public int createMethod(Method method);

   /**
    * This method updates a method in the database.
    *
    * @param method
    *          - The method to update.
    */
   public void updateMethod(Method method);

   /**
    * This method removes a method from the database.
    *
    * @param methodId
    *          - The identifier of the method to remove.
```

```
    */
   public void deleteMethod(int methodId);

   /**
    * This method fetches the list of methods from the database.
    *
    * @return A list of Method objects if some methods have been found. Else return null.
    */
   public List<Method> listMethods();

   /**
    * This method fetches a method in the database based on its identifier.
    *
    * @param methodId
    *          - The identifier of the method to fetch.
    * @return A Method object if found. Else return null.
    */
   public Method getMethodById(int methodId);

   /**
    * This method fetches a method in the database based on its name.
    *
    * @param methodName
    *          - The name of the method to fetch.
    * @return A Method object if found. Else return null.
    */
   public Method getMethodByName(String methodName);

}
```

2.3 PracticeDao.java

```
package com.unamur.dao;

import java.util.List;
import com.unamur.entity.Practice;

public interface PracticeDao {

   /**
    * This method creates a practice in the database.
    *
    * @param practice
    *          - The practice to create.
    * @return The identifier of the newly created practice.
    */
   public int createPractice(Practice practice);

   /**
    * This method updates a practice in the database.
    *
    * @param practice
    *          - The practice to update.
    */
   public void updatePractice(Practice practice);

   /**
    * This method removes a practice from the database.
```

```
     *
     * @param practiceId
     *          - The identifier of the practice to remove.
     */
    public void deletePractice(int practiceId);

    /**
     * This method fetches the list of practices from the database.
     *
     * @return A list of Practice objects if some practices have been found. Else return null.
     */
    public List<Practice> listPractices();

    /**
     * This method fetches a practice in the database based on its identifier.
     *
     * @param practiceId
     *          - The identifier of the practice to fetch.
     * @return A Practice object if found. Else return null.
     */
    public Practice getPracticeById(int practiceId);

    /**
     * This method fetches a practice in the database based on its name.
     *
     * @param practiceName
     *          - The name of the practice to fetch.
     * @return A Practice object if found. Else return null.
     */
    public Practice getPracticeByName(String name);

}
```

2.4 MethodCommentDao.java

```
package com.unamur.dao;

import com.unamur.entity.MethodComment;

public interface MethodCommentDao {

    /**
     * This method creates a comment linked to a method in the database.
     *
     * @param comment
     *          - The comment to create.
     */
    public void createMethodComment(MethodComment comment);

    /**
     * This method removes a comment linked to a method from the database.
     *
     * @param commentId
     *          - The identifier of the comment to delete.
     */
    public void deleteMethodComment(int commentId);
}
```

2.5 PracticeCommentDao.java

```
package com.unamur.dao;

import com.unamur.entity.PracticeComment;

public interface PracticeCommentDao {

   /**
    * This method creates a comment linked to a practice in the database.
    *
    * @param comment
    *          - The comment to create.
    */
   public void createPracticeComment(PracticeComment comment);

   /**
    * This method removes a comment linked to a practice from the database.
    *
    * @param commentId
    *          - The identifier of the comment to delete.
    */
   public void deletePracticeComment(int commentId);
}
```

3 DAO - Implémentations

3.1 UserDaoImpl.java

```
package com.unamur.dao.impl;

import java.util.List;

import org.apache.log4j.Logger;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.unamur.dao.UserDao;
import com.unamur.entity.User;

@Repository
public class UserDaoImpl implements UserDao {

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(UserDaoImpl.class);

   @Autowired
   private SessionFactory sessionFactory;

   /**
    * {@inheritDoc} @
    */
   @Override
   public void createUser(User user) {

         log.debug("Start method createUser");

         /* Insert the user in the database. */
         sessionFactory.getCurrentSession().save(user);

         log.debug("End method createUser");
   }

   /**
    * {@inheritDoc} @
    */
   @Override
   public User getUserByEmail(String email) {

         log.debug("Start method getUserByEmail");
         log.debug("Loof for user with email : " + email);

         /* Get the session bound to the context. */
         Session session = sessionFactory.getCurrentSession();

         /* Look for the user in the database. */
         @SuppressWarnings("unchecked")
         List<User> list = session.createQuery(
               "from User u where u.email = :email").setParameter("email", email).list();

         /* Return the user if found. Else return null. */
```

```
        User user = list.size() > 0 ? (User) list.get(0) : null;

        log.debug(list.size() > 0 ? "User has been found in the database" : "User has not been found in the
database");
        log.debug("End method getUserByEmail");

        return user;
   }

}
```

3.2 MethodDaoImpl.java

```
package com.unamur.dao.impl;

import java.util.List;

import org.apache.log4j.Logger;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.unamur.dao.MethodDao;
import com.unamur.entity.Method;

@Repository
public class MethodDaoImpl implements MethodDao {

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(MethodDaoImpl.class);

   @Autowired
   private SessionFactory sessionFactory;

   /**
    * {@inheritDoc} @
    */
   @Override
   public int createMethod(Method method) {

        log.debug("Start method createMethod");

        /*
         * Insert the method in the database and return the id of the newly
         * created record.
         */
        int generatedId = (Integer) sessionFactory.getCurrentSession().save(method);

        log.debug("Method has been created with id : " + generatedId);
        log.debug("End method createMethod");

        return generatedId;
   }

   /**
    * {@inheritDoc} @
    */
```

```
@Override
public void updateMethod(Method method) {

	log.debug("Start method updateMethod");
	log.debug("Update method with id : " + method.getId() + " and name : " + method.getName());

	/*
	 * Update the method in the database.
	 */
	sessionFactory.getCurrentSession().update(method);

	log.debug("End method updateMethod");
}

/**
 * {@inheritDoc} @
 */
@Override
public void deleteMethod(int methodId) {

	log.debug("Start method deleteMethod");
	log.debug("Delete method with id : " + methodId);

	Method method = (Method) sessionFactory.getCurrentSession().load(Method.class, methodId);

	if (method != null) {

	   /* Remove the method from the database. */
	   sessionFactory.getCurrentSession().delete(method);
	}

	log.debug("End method deleteMethod");

}

/**
 * {@inheritDoc} @
 */
@SuppressWarnings("unchecked")
@Override
public List<Method> listMethods() {

	log.debug("Start method listMethods");

	/*
	 * Get and return the list of methods from the database.
	 */
	List <Method> methods = sessionFactory.getCurrentSession()
			.createQuery("from Method m order by m.name").list();

	log.info(methods.size() + " methods have been found");
	log.debug("End method listMethods");

	return methods;
}
```

```
	/**
	 * {@inheritDoc} @
	 */
	@Override
	public Method getMethodById(int methodId) {

		log.debug("Start method getMethodById");
		log.debug("Fetch method with id : " + methodId);

		/* Get the session bound to the context. */
		Session session = sessionFactory.getCurrentSession();

		/* Look for the method in the database. */
		@SuppressWarnings("unchecked")
		List<Method> list = session
			.createQuery("from Method m where m.id = :id")
			.setParameter("id", methodId).list();

		/* Return the method if found. Else return null. */
		Method method = list.size() > 0 ? (Method) list.get(0) : null;

		log.debug(list.size() > 0 ? "Method has been found in the database" : "Method has not been found in
the database");
		log.debug("End method getMethodById");

		return method;
	}

	/**
	 * {@inheritDoc} @
	 */
	@Override
	public Method getMethodByName(String name) {

		log.debug("Start method getMethodByName");
		log.debug("Fetch method with name : " + name);

		/* Get the session bound to the context. */
		Session session = sessionFactory.getCurrentSession();

		/* Look for the method in the database. */
		@SuppressWarnings("unchecked")
		List<Method> list = session
			.createQuery("from Method m where m.name = :name")
			.setParameter("name", name).list();

		/* Return the method if found. Else return null. */
		Method method = list.size() > 0 ? (Method) list.get(0) : null;

		log.debug(list.size() > 0 ? "Method has been found in the database" : "Method has not been found in
the database");
		log.debug("End method getMethodByName");

		return method;
	}
}
```

3.3 PracticeDaoImpl.java

```
package com.unamur.dao.impl;

import java.util.List;

import org.apache.log4j.Logger;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.unamur.dao.PracticeDao;
import com.unamur.entity.Practice;

@Repository
public class PracticeDaoImpl implements PracticeDao {

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(PracticeDaoImpl.class);

   @Autowired
   private SessionFactory sessionFactory;

   /**
    * {@inheritDoc} @
    */
   @Override
   public int createPractice(Practice practice) {

         log.debug("Start method createPractice");

         /*
          * Insert the practice in the database and return the id of the newly
          * created record.
          */
         int generatedId = (Integer) sessionFactory.getCurrentSession().save(practice);

         log.debug("Practice has been created with id : " + generatedId);
         log.debug("End method createPractice");

         return generatedId;
   }

   /**
    * {@inheritDoc} @
    */
   @Override
   public void updatePractice(Practice practice) {

         log.debug("Start method updatePractice");
         log.debug("Update practice with id : " + practice.getId() + " and name : " + practice.getName());

         /*
          * Update the practice in the database.
          */
         sessionFactory.getCurrentSession().update(practice);
```

```
        log.debug("End method updatePractice");
}

/**
 * {@inheritDoc} @
 */
@Override
public void deletePractice(int practiceId) {

        log.debug("Start method deletePractice");
        log.debug("Delete method with id : " + practiceId);

        Practice practice = (Practice) sessionFactory.getCurrentSession().load(Practice.class, practiceId);

        if (practice != null) {

           /* Remove the practice from the database. */
           sessionFactory.getCurrentSession().delete(practice);
        }

        log.debug("End method deletePractice");
}

/**
 * {@inheritDoc} @
 */
@SuppressWarnings("unchecked")
@Override
public List<Practice> listPractices() {

        log.debug("Start method listPractices");

        /*
         * Get and return the list of practices from the database.
         */
        List <Practice> practices = sessionFactory.getCurrentSession()
                .createQuery("from Practice p order by p.name").list();

        log.info(practices.size() + " practices have been found");
        log.debug("End method listPractices");

        return practices;
}

/**
 * {@inheritDoc} @
 */
@Override
public Practice getPracticeById(int practiceId) {

        log.debug("Start method getPracticeById");
        log.debug("Fetch practice with id : " + practiceId);

        /* Get the session bound to the context. */
        Session session = sessionFactory.getCurrentSession();

        /* Look for the practice in the database. */
```

```
		@SuppressWarnings("unchecked")
		List<Practice> list = session
				.createQuery("from Practice p where p.id = :id")
				.setParameter("id", practiceId).list();

		/* Return the practice if found. Else return null. */
		Practice practice = list.size() > 0 ? (Practice) list.get(0) : null;

		log.debug(list.size() > 0 ? "Practice has been found in the database" : "Practice has not been found in
the database");
		log.debug("End method getPracticeById");

		return practice;
	}

	/**
	 * {@inheritDoc} @
	 */
	@Override
	public Practice getPracticeByName(String name) {

		log.debug("Start method getPracticeByName");
		log.debug("Fetch practice with name : " + name);

		/* Get the session bound to the context. */
		Session session = sessionFactory.getCurrentSession();

		/* Look for the practice in the database. */
		@SuppressWarnings("unchecked")
		List<Practice> list = session
				.createQuery("from Practice p where p.name = :name")
				.setParameter("name", name).list();

		/* Return the practice if found. Else return null. */
		Practice practice = list.size() > 0 ? (Practice) list.get(0) : null;

		log.debug(list.size() > 0 ? "Practice has been found in the database" : "Practice has not been found in
the database");
		log.debug("End method getPracticeByName");

		return practice;
	}
}
```

3.4 MethodCommentDaoImpl.java

```
package com.unamur.dao.impl;

import org.apache.log4j.Logger;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.unamur.dao.MethodCommentDao;
import com.unamur.entity.MethodComment;

@Repository
public class MethodCommentDaoImpl implements MethodCommentDao {
```

```
  /* Logger */
  public static org.apache.log4j.Logger log = Logger.getLogger(MethodCommentDaoImpl.class);

  @Autowired
  private SessionFactory sessionFactory;

  /**
   * {@inheritDoc} @
   */
  @Override
  public void createMethodComment(MethodComment comment) {

        log.debug("Start method createMethodComment");

        /* Insert the comment in the database. */
        sessionFactory.getCurrentSession().save(comment);

        log.debug("End method createMethodComment");
  }

  /**
   * {@inheritDoc} @
   */
  @Override
  public void deleteMethodComment(int commentId) {

        log.debug("Start method deleteMethodComment");
        log.debug("Delete comment with id : " + commentId);

        MethodComment methodComment = (MethodComment) sessionFactory
              .getCurrentSession().load(MethodComment.class, commentId);

        if (methodComment != null) {

          /* Remove the comment from the database. */
          sessionFactory.getCurrentSession().delete(methodComment);
        }

        log.debug("End method deleteMethodComment");

  }

}
```

3.5 PracticeCommentDaoImpl.java

```
package com.unamur.dao.impl;

import org.apache.log4j.Logger;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.unamur.dao.PracticeCommentDao;
import com.unamur.entity.PracticeComment;

@Repository
```

```
public class PracticeCommentDaoImpl implements PracticeCommentDao {

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(PracticeCommentDaoImpl.class);

   @Autowired
   private SessionFactory sessionFactory;

   /**
    * {@inheritDoc} @
    */
   @Override
   public void createPracticeComment(PracticeComment comment) {

           log.debug("Start method createPracticeComment");

           /* Insert the comment in the database. */
           sessionFactory.getCurrentSession().save(comment);

           log.debug("End method createPracticeComment");
   }

   /**
    * {@inheritDoc} @
    */
   @Override
   public void deletePracticeComment(int commentId) {

           log.debug("Start method deletePracticeComment");
           log.debug("Delete comment with id : " + commentId);

           PracticeComment practiceComment = (PracticeComment) sessionFactory
                   .getCurrentSession().load(PracticeComment.class, commentId);

           if (practiceComment != null) {

              /* Remove the comment from the database. */
              sessionFactory.getCurrentSession().delete(practiceComment);
           }

           log.debug("End method deletePracticeComment");

   }

}
```

4 Services - Interfaces

4.1 UserService.java

```
package com.unamur.service;

import com.unamur.EntryDuplicatedException;
import com.unamur.entity.User;

public interface UserService {

   /**
    * This method creates a user. If the user already exists, an
    * exception is returned. Else the user is created.
    *
    * @param user
    *          - The user to create.
    * @exception EntryDuplicatedException
    *             - This exception is returned if the user already exists.
    */
   public void createUser(User user) throws EntryDuplicatedException;
}
```

4.2 MethodService.java

```
package com.unamur.service;

import java.util.List;

import org.springframework.security.access.prepost.PreAuthorize;

import com.unamur.EntryDuplicatedException;
import com.unamur.entity.Method;
import com.unamur.entity.MethodComment;

public interface MethodService {

   /**
    * This method creates a method.
    *
    * @param method
    *          - The method to create.
    * @return The identifier of the newly created method.
    * @exception EntryDuplicatedException
    *             - This exception is returned if the method already exists.
    */
   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
   public int createMethod(Method method) throws EntryDuplicatedException;

   /**
    * This method updates a method.
    *
    * @param method
    *          - The method to update.
    */
   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
   public void updateMethod(Method method);

   /**
```

```
    * This method deletes a method.
    *
    * @param methodId
    *          - The identifier of the method to delete.
    */
   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
   public void deleteMethod(int methodId);

   /**
    * This method fetches the list of methods.
    *
    * @return A list of Method objects if some methods have been found. Else return null.
    */
   public List<Method> listMethods();

   /**
    * This method fetches a method in the database based on its identifier.
    *
    * @param methodId
    *          - The identifier of the method to fetch.
    * @return A Method object if found. Else return null.
    */
   public Method getMethodById(int methodId);

   /**
    * This method creates a comment linked to a method.
    *
    * @param comment
    *          - The comment to create.
    */
   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR', 'ROLE_VISITOR')")
   public void createMethodComment(MethodComment comment);

   /**
    * This method removes a comment linked to a method.
    *
    * @param commentId
    *          - The identifier of the comment.
    */
   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR', 'ROLE_VISITOR')")
   public void deleteMethodComment(int commentId);
}
```

4.3 PracticeService.java

```
package com.unamur.service;

import java.util.List;

import org.springframework.security.access.prepost.PreAuthorize;

import com.unamur.EntryDuplicatedException;
import com.unamur.entity.Practice;
import com.unamur.entity.PracticeComment;

public interface PracticeService {

   /**
```

```
 * This method creates a practice.
 *
 * @param practice
 *          - The practice to create.
 * @return The identifier of the newly created practice.
 * @exception EntryDuplicatedException
 *              - This exception is returned if the practice already exists.
 */
@PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
public int createPractice(Practice practice) throws EntryDuplicatedException;

/**
 * This method updates a practice.
 *
 * @param practice
 *          - The practice to update.
 */
@PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
public void updatePractice(Practice practice);

/**
 * This method deletes a practice.
 *
 * @param practiceId
 *          - The identifier of the practice to delete.
 */
@PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
public void deletePractice(int practiceId);

/**
 * This method fetches the list of practices.
 *
 * @return A list of Practice objects if some practices have been found. Else return null.
 */
public List<Practice> listPractices();

/**
 * This method fetches a practice in the database based on its identifier.
 *
 * @param practiceId
 *          - The identifier of the practice to fetch.
 * @return A Practice object if found. Else return null.
 */
public Practice getPracticeById(int practiceId);

/**
 * This method creates a comment linked to a practice.
 *
 * @param comment
 *          - The comment to create.
 */
@PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR', 'ROLE_VISITOR')")
public void createPracticeComment(PracticeComment comment);

/**
 * This method removes a comment linked to a practice.
 *
```

```
    * @param commentId
    *          - The identifier of the comment.
    */
   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR', 'ROLE_VISITOR')")
   public void deletePracticeComment(int commentId);

}
```

5 Services - Implémentations

5.1 UserServiceImpl.java

```
package com.unamur.service.impl;

import javax.transaction.Transactional;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import com.unamur.EntryDuplicatedException;
import com.unamur.dao.UserDao;
import com.unamur.entity.Role;
import com.unamur.entity.User;
import com.unamur.service.UserService;

@Service
public class UserServiceImpl implements UserService {

   /* Constant : Role assigned by default to newly created users. */
   public static final int ROLE_DEFAULT = 1;

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(UserServiceImpl.class);

   @Autowired
   private UserDao userDao;

   /**
    * {@inheritDoc} @
    */
   @Transactional
   public void createUser(User user) throws EntryDuplicatedException {

         log.debug("Start method createUser");

         /* Check if the user already exists. */
         if (userDao.getUserByEmail(user.getEmail()) != null) {

            /* Return an exception if the user already exists. */
            log.info("User with email : " + user.getEmail() + " already exists");
            throw new EntryDuplicatedException();
         }

         /* If the user does not exist yet. */
         else {

            log.debug("User does not exist");
            PasswordEncoder passwordEncoder = new BCryptPasswordEncoder();

            Role role = new Role();

            /* Encode the user password with bcrypt */
            user.setPassword(passwordEncoder.encode(user.getPassword()));
```

```
            /* Assign the default user role. */
            role.setId(ROLE_DEFAULT);
            user.setRole(role);

            /* Insert the user in the database. */
            userDao.createUser(user);

            log.debug("End method createUser");
        }
    }
}
```

5.2 MethodServiceImpl.java

```
package com.unamur.service.impl;

import java.util.List;

import javax.transaction.Transactional;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;

import com.unamur.EntryDuplicatedException;
import com.unamur.dao.MethodCommentDao;
import com.unamur.dao.MethodDao;
import com.unamur.dao.UserDao;
import com.unamur.entity.Method;
import com.unamur.entity.MethodComment;
import com.unamur.entity.User;
import com.unamur.service.MethodService;

@Service
public class MethodServiceImpl implements MethodService {

    /* Logger */
    public static org.apache.log4j.Logger log = Logger.getLogger(MethodServiceImpl.class);

    @Autowired
    private MethodDao methodDao;

    @Autowired
    private MethodCommentDao methodCommentDao;

    @Autowired
    private UserDao userDao;

    /**
     * {@inheritDoc} @
     */
    @Transactional
    public int createMethod(Method method) throws EntryDuplicatedException {

            log.debug("Start method createMethod");
```

```
        int generatedId;

        /* Check if the method already exists. */
        if (methodDao.getMethodByName(method.getName()) != null){
           /*
            * Return an exception EntryDuplicatedException if the method
            * already exists.
            */
           log.info("Method : " + method.getName() + " already exists");
           throw new EntryDuplicatedException();
}

        /* Save the method. */
        generatedId = methodDao.createMethod(method);

                log.info("Method " + method.getName()
                        + " has been created. Generated ID is " + generatedId);
                log.debug("End method createMethod");

        return generatedId ;
}

/**
 * {@inheritDoc} @
 */
@Transactional
public void deleteMethod(int methodId) {

        log.debug("Start method deleteMethod");
        log.debug("Delete method with id : " + methodId);

        /* Delete the method. */
        methodDao.deleteMethod(methodId);

        log.debug("End method deleteMethod");
}

/**
 * {@inheritDoc} @
 */
@Transactional
public List<Method> listMethods() {

        log.debug("Start method listMethods");

        /* Get and return the list of methods. */

        List <Method> methods = methodDao.listMethods();

        log.info(methods.size() + " methods have been found");
        log.debug("End method listMethods");

        return methods;
}

/**
```

```
 * {@inheritDoc} @
 */
@Transactional
public void updateMethod(Method method) {

	log.debug("Start method updateMethod");
	log.debug("Update method with id : " + method.getId() + " and name : " + method.getName());

	/* Update the method. */
	methodDao.updateMethod(method);

	log.debug("End method updateMethod");
}

/**
 * {@inheritDoc} @
 */
@Transactional
public Method getMethodById(int methodId) {

	log.debug("Start method getMethodById");
	log.debug("Fetch method with id : " + methodId);

	/* Get & return the method. */
	Method method = methodDao.getMethodById(methodId);

	log.debug("End method getMethodById");

	return method;
}

/**
 * {@inheritDoc} @
 */
@Transactional
public void createMethodComment(MethodComment comment) {

	log.debug("Start method createMethodComment");

	/* Get the Security Principal. */
	Authentication auth = SecurityContextHolder.getContext().getAuthentication();

	log.debug("Security principal is " + auth.getName());

	/* Get authenticated user information. */
	User author = userDao.getUserByEmail(auth.getName());

	/* Set the authenticated user as the author of the comment. */
	comment.setAuthor(author);

	/* Save the comment. */
	methodCommentDao.createMethodComment(comment);

	log.debug("End method createMethodComment");
}

/**
```

```
    * {@inheritDoc} @
    */
   @Transactional
   public void deleteMethodComment(int commentId) {

           log.debug("Start method deleteMethodComment");
           log.debug("Delete method with id : " + commentId);

           /* Delete the comment. */
           methodCommentDao.deleteMethodComment(commentId);

           log.debug("End method deleteMethodComment");
   }

}
```

5.3 PracticeServiceImpl.java

```
package com.unamur.service.impl;

import java.util.List;

import javax.transaction.Transactional;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;

import com.unamur.EntryDuplicatedException;
import com.unamur.dao.PracticeCommentDao;
import com.unamur.dao.PracticeDao;
import com.unamur.dao.UserDao;
import com.unamur.entity.Practice;
import com.unamur.entity.PracticeComment;
import com.unamur.entity.User;
import com.unamur.service.PracticeService;

@Service
public class PracticeServiceImpl implements PracticeService {

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(PracticeServiceImpl.class);

   @Autowired
   private PracticeDao practiceDao;

   @Autowired
   private PracticeCommentDao practiceCommentDao;

   @Autowired
   private UserDao userDao;

   /**
    * {@inheritDoc} @
    */
   @Transactional
```

```
public int createPractice(Practice practice) throws EntryDuplicatedException {

	log.debug("Start method createPractice");

	int generatedId;

	/* Check if the practice already exists. */
	if (practiceDao.getPracticeByName(practice.getName()) != null) {
	   /*
	    * Return an exception EntryDuplicatedException if the practice already exists.
	    */
	   log.info("Practice : " + practice.getName() + " already exists");
	   throw new EntryDuplicatedException();
	}

	/* Save the practice. */
	generatedId = practiceDao.createPractice(practice);

	log.info("Practice " + practice.getName() + " has been created. Generated ID is " + generatedId);
	log.debug("End method createPractice");

	return generatedId;
}

/**
 * {@inheritDoc} @
 */
@Transactional
public void deletePractice(int practiceId) {

	log.debug("Start method deletePractice");
	log.debug("Delete practice with id : " + practiceId);

	/* Delete the practice. */
	practiceDao.deletePractice(practiceId);

	log.debug("End method deletePractice");
}

/**
 * {@inheritDoc} @
 */
@Transactional
public List<Practice> listPractices() {

	log.debug("Start method listPractices");

	/* Get and return the list of practices. */
	List <Practice> practices = practiceDao.listPractices();

	log.info(practices.size() + " practices have been found");
	log.debug("End method listPractices");

	return practices;
}

/**
```

```
 * {@inheritDoc} @
 */
@Transactional
public void updatePractice(Practice practice) {

	log.debug("Start method updatePractice");
	log.debug("Update practice with id : " + practice.getId() + " and name : " + practice.getName());

	/* Update the practice. */
	practiceDao.updatePractice(practice);

	log.debug("End method updatePractice");
}

/**
 * {@inheritDoc} @
 */
@Transactional
public Practice getPracticeById(int practiceId) {

	log.debug("Start method getPracticeById");
	log.debug("Fetch practice with id : " + practiceId);

	/* Get & return the practice. */
	Practice practice = practiceDao.getPracticeById(practiceId);

	log.debug("End method getPracticeById");

	return practice;
}

/**
 * {@inheritDoc} @
 */
@Transactional
public void createPracticeComment(PracticeComment comment) {

	log.debug("Start method createPracticeComment");

	/* Get the Security Principal. */
	Authentication auth = SecurityContextHolder.getContext().getAuthentication();

	log.debug("Security principal is " + auth.getName());

	/* Get authenticated user information. */
	User author = userDao.getUserByEmail(auth.getName());

	/* Set the authenticated user as the author of the comment. */
	comment.setAuthor(author);

	/* Save the comment. */
	practiceCommentDao.createPracticeComment(comment);

	log.debug("End method createPracticeComment");
}

/**
```

```
    * {@inheritDoc} @
    */
   @Transactional
   public void deletePracticeComment(int commentId) {

           log.debug("Start method deletePracticeComment");
           log.debug("Delete practice with id : " + commentId);

           /* Delete the comment. */
           practiceCommentDao.deletePracticeComment(commentId);

           log.debug("End method deletePracticeComment");
   }

}
```

6 Contrôleurs

6.1 WebController.java

```
package com.unamur.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class WebController {

	@RequestMapping(value = "/home", method = RequestMethod.GET)
	public String index() {

	   return "/home";
	}

}
```

6.2 UserController.java

```
package com.unamur.controller;

import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.unamur.EntryDuplicatedException;
import com.unamur.entity.User;
import com.unamur.service.UserService;

@Controller
public class UserController {

   @Autowired
   private UserService userService;

   @ModelAttribute("user")
   public User getUserForm() {
	return new User();
   }

   @RequestMapping(value = "/subscribe", method = RequestMethod.GET)
   public String subscribe(@SuppressWarnings("rawtypes") Map model) {

	/* Load subscription page. */
	return "addUser";
   }
```

```
   @RequestMapping(value = "/subscribe", method = RequestMethod.POST)
   public String subscribe(@ModelAttribute("user") @Valid User user,
            BindingResult result,
            RedirectAttributes redirectAttributes) {

         if (result.hasErrors()) {
            /*
             * If there are some validation errors. Then, redirect the user back to the subscription page.
             */
            return "addUser";
         }

         try {
            /* Create user. */
            userService.createUser(user);
         } catch (EntryDuplicatedException error) {

            /* If the user already exists. Then, return an error message to the user.*/
            result.rejectValue("email", "email.already.exists");

            return "addUser";
         }

         /* Post success message. */
         redirectAttributes.addFlashAttribute("alertSuccessMessage", "alert.message.subscribe.success");

         /* Redirect the user to the login page. */
         return "redirect:/login";
   }

}
```

6.3 LoginController.java

```
package com.unamur.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class LoginController {

   /**
    * Login
    * */
   @RequestMapping(value = "/login", method = RequestMethod.GET)
   public String login() {

         return "login";

   }

}
```

6.4 MethodController.java

```
package com.unamur.controller;

import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.unamur.EntryDuplicatedException;
import com.unamur.PracticePropertyEditor;
import com.unamur.entity.Method;
import com.unamur.entity.MethodComment;
import com.unamur.entity.Practice;
import com.unamur.service.MethodService;
import com.unamur.service.PracticeService;

@Controller
public class MethodController {

   @Autowired
   private MethodService methodService;

   @Autowired
   private PracticeService practiceService;

   @Autowired
   private MessageSource messageSource;

   @ModelAttribute("method")
   public Method getMethodForm() {
         return new Method();
   }

   @ModelAttribute("methodComment")
   public MethodComment getMethodCommentForm() {
         return new MethodComment();
   }

   @InitBinder
   public void initBinder(WebDataBinder binder)
   {
      binder.registerCustomEditor(Practice.class, new PracticePropertyEditor(practiceService));
   }
```

```
@RequestMapping(value = "/methods")
public String listMethods(Map<String, Object> map) {

	/* Get the list of methods */

	map.put("method", new Method());
	map.put("methodList", methodService.listMethods());

	return "listMethods";
}

@RequestMapping(value = "/secure/addMethod", method = RequestMethod.GET)
public String addMethod(@SuppressWarnings("rawtypes") Map model) {

	/* Load the creation page */

	return "addMethod";

}

@RequestMapping(value = "/secure/addMethod", method = RequestMethod.POST)
public String addMethod(
	   @ModelAttribute("method") @Valid Method method,
	   BindingResult result,
	   RedirectAttributes redirectAttributes) {

	int generatedMethodId;

	/* Check if there is validation errors in the form */

	if (result.hasErrors()) {

	   /*
	    * If there are errors. Then, redirect the user back to the creation of the method.
	    */

	   return "addMethod";
	}

	/* Create method */
	try {
	   generatedMethodId = methodService.createMethod(method);

	} catch (EntryDuplicatedException error) {
	   result.rejectValue("name", "method.already.exists");

	   return "addMethod";

	}

	/* Post success message */
	redirectAttributes.addFlashAttribute("alertSuccessMessage", "alert.message.method.add.success");

	/* Redirect the user to the edition of a method */
	return "redirect:/method/details/" + generatedMethodId;
}
```

```
@RequestMapping("/secure/deleteMethod/{methodId}")
public String deleteMethod(
        @PathVariable("methodId") int methodId,
        RedirectAttributes redirectAttributes) {

    /* Delete method */
    methodService.deleteMethod(methodId);

    /* Post success message */
    redirectAttributes.addFlashAttribute("alertSuccessMessage","alert.message.method.delete.success");

    /* Redirect the user to the list of methods */
    return "redirect:/methods";
}

@RequestMapping(value = "/method/details/postMethodComment", method = RequestMethod.POST)
public String postMethodComment(
        @ModelAttribute("methodComment") @Valid MethodComment comment,
        BindingResult result) {

    /* Create comment */
    methodService.createMethodComment(comment);

    return "redirect:/method/details/" + comment.getMethod().getId() + "#comments";
}

@RequestMapping(value = "/method/details/{methodId}", method = RequestMethod.GET)
public String editMethod(@PathVariable("methodId") int methodId, Model model) {

    /* Get method details */
    model.addAttribute("method", methodService.getMethodById(methodId));

    /* Get all the existing methods */
    model.addAttribute("existingMethodList", methodService.listMethods());

    /* Get all the existing practices */
    model.addAttribute("existingPracticeList", practiceService.listPractices());

    return "detailsMethod";
}

@RequestMapping(value = "/method/details/{methodId}", method = RequestMethod.POST)
public String editMethod(
        @ModelAttribute("method") @Valid Method method,
        BindingResult result,
        RedirectAttributes redirectAttributes, Model model) {

    /* Check if there is validation errors in the form */
    if (result.hasErrors()) {

        /*
         * If there are errors. Then, redirect the user back to the modification of the method
         */
        return "detailsMethod";
    }
```

```
        /* Update practice */

        methodService.updateMethod(method);

        /* Post success message */

        redirectAttributes.addFlashAttribute("alertSuccessMessage", "alert.message.method.edit.success");

        return "redirect:/method/details/" + method.getId();
    }

    @RequestMapping(value = "/deleteMethodComment/", method = RequestMethod.GET)
    @ResponseBody
    public String deleteMethodComment(@RequestParam int commentId) {

        /* Delete comment */
        methodService.deleteMethodComment(commentId);

        return "success";
    }

}
```

6.5 PracticeController.java

```
package com.unamur.controller;

import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.unamur.EntryDuplicatedException;
import com.unamur.entity.Practice;
import com.unamur.entity.PracticeComment;
import com.unamur.service.PracticeService;

@Controller
public class PracticeController {

    @Autowired
    private PracticeService practiceService;

    @ModelAttribute("practice")
    public Practice getForm() {
        return new Practice();
```

```
}

@ModelAttribute("practiceComment")
public PracticeComment getFormPracticeComment() {
	return new PracticeComment();
}

@RequestMapping(value = "/practices", method = RequestMethod.GET)
public String listPractices(Model model) {

	model.addAttribute("practice", new Practice());

	/* Get the list of practices */
	model.addAttribute("practiceList", practiceService.listPractices());

	return "listPractices";
}

@RequestMapping(value = "/practice/details/{practiceId}", method = RequestMethod.GET)
public String editPractice(
	   @PathVariable("practiceId") int practiceId, Model model) {

	/* Get practice details */
	model.addAttribute("practice", practiceService.getPracticeById(practiceId));

	return "detailsPractice";
}

@PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
@RequestMapping(value = "/practice/details/{practiceId}", method = RequestMethod.POST)
public String editPractice(
	   @ModelAttribute("practice") @Valid Practice practice,
	   BindingResult result, RedirectAttributes redirectAttributes, Model model) {

	/* Check if there is validation errors in the form */

	if (result.hasErrors()) {

	   /* If there are errors. Then, redirect the user back to the
	    modification of the practice */

	   return "detailsPractice";
	}

	/* Update practice */

	practiceService.updatePractice(practice);

	/* Post success message */

	redirectAttributes.addFlashAttribute("alertSuccessMessage", "alert.message.practice.edit.success");

	return "redirect:/practice/details/" + practice.getId();
}

@PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
@RequestMapping(value = "/secure/deletePractice/{practiceId}", method = RequestMethod.GET)
```

```
   public String deletePractice(@PathVariable("practiceId") int practiceId, RedirectAttributes
redirectAttributes) {

        /* Delete the practice */

        practiceService.deletePractice(practiceId);

        /* Post success message */

        redirectAttributes.addFlashAttribute("alertSuccessMessage","alert.message.practice.delete.success");

        /* Redirect the user to the list of practices */

        return "redirect:/practices";
   }

   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
   @RequestMapping(value = "/secure/addPractice", method = RequestMethod.GET)
   public String showPracticeForm(@SuppressWarnings("rawtypes") Map model) {
        return "addPractice";
   }

   @PreAuthorize("hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')")
   @RequestMapping(value = "/secure/addPractice", method = RequestMethod.POST)
   public String addPractice(
           @ModelAttribute("practice") @Valid Practice practice,
           BindingResult result, RedirectAttributes redirectAttributes) {

        int generatedPracticeId;

        /* Check if there is validation errors in the form */

        if (result.hasErrors()) {

           /*
            * If there are validation errors. Then, redirect the user back to
            * the creation of the practice
            */

           return "addPractice";
        }

        /* Create practice */

        try {
           generatedPracticeId = practiceService.createPractice(practice);

        } catch (EntryDuplicatedException error) {
           result.rejectValue("name", "practice.already.exists");

           /* If the practice already exists. Then, redirect the user back to the
            creation of the practice */

           return "addPractice";
        }

        /* Post success message */
```

```
        redirectAttributes.addFlashAttribute("alertSuccessMessage","alert.message.practice.add.success");

        return "redirect:/practice/details/" + generatedPracticeId;

    }

    @RequestMapping(value = "/practice/details/postPracticeComment", method = RequestMethod.POST)
    public String postPracticeComment(
            @ModelAttribute("practiceComment") @Valid PracticeComment comment,
            BindingResult result) {

        /* Check if there is validation errors in the form */
        if (result.hasErrors()) {
            return "consultPractice";
        }

        /* Create comment */
        practiceService.createPracticeComment(comment);

        return "redirect:/practice/details/" + comment.getPractice().getId() + "#comments";
    }

    @RequestMapping(value="/deletePracticeComment/", method= RequestMethod.GET)
    @ResponseBody
    public String deletePracticeComment(@RequestParam int commentId) {

        /* Delete comment */
        practiceService.deletePracticeComment(commentId);

        return "success";
    }

}
```

7 Autres classes Java

7.1 EntryDuplicatedException.java

```
package com.unamur;
/**
 * Exception to be returned in case of non authorized duplicates records.
 * */
@SuppressWarnings("serial")
public class EntryDuplicatedException extends Exception {

   public EntryDuplicatedException() {
   }

   public EntryDuplicatedException(String message) {
          super(message);
   }

   public EntryDuplicatedException(Throwable cause) {
          super(cause);
   }

   public EntryDuplicatedException(String message, Throwable cause) {
          super(message, cause);
   }
}
```

7.2 PracticePropertyEditor.java

```
package com.unamur;
import java.beans.PropertyEditorSupport;
import org.apache.log4j.Logger;
import com.unamur.entity.Practice;
import com.unamur.service.PracticeService;

public class PracticePropertyEditor extends PropertyEditorSupport {
   private PracticeService practiceService;

   /* Logger */
   public static org.apache.log4j.Logger log = Logger.getLogger(PracticePropertyEditor.class);

   public PracticePropertyEditor(PracticeService practiceService) {
          this.practiceService = practiceService;
   }

   @Override
   public void setAsText(String id) throws IllegalArgumentException {

          log.debug("Start method setAsText");
          log.debug("Id : " + id);

          Practice practice = practiceService.getPracticeById(Integer.valueOf(id));

          setValue(practice);

          log.debug("End method setAsText");
   }
}
```

8 Fichiers JSP

8.1 index.jsp

```
<!-- Redirect the user to the home page. -->
<jsp:forward page="home.html"></jsp:forward>
```

8.2 home.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!DOCTYPE html>

<html>

<head>
 <title>
  <spring:message code="title.home" />
 </title>

 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>
</head>

<body>

 <!-- Navigation area -->
 <jsp:include page="navigation.jsp" />

 <div class="container">

  <!-- Breadcrumb area -->
  <ol class="breadcrumb">
   <li class="active">
    <spring:message code="breadcrumb.home" />
   </li>
  </ol>

  <!-- If the user logged in successfully. Then notify him. -->
  <c:if test="${param.login != null}">

   <div class="alert alert-info alert-dismissible" role="alert">
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
        <span aria-hidden="true">&times;</span>
    </button>
    <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
    <spring:message code="alert.message.login.success"/>
   </div>

  </c:if>

  <div class="jumbotron">
```

```
    <h1> <spring:message code="text.home.welcome"/>  </h1>
    <p>  <spring:message code="text.home.baseline"/> </p>
   </div>

  </div>
</body>

</html>
```

8.3 login.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<!DOCTYPE html>
<html>

<head>

 <title>
  <spring:message code="title.login" />
 </title>

 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>

</head>

<body>

 <spring:message code="placeholder.user.email" var="placeHolderEmail" />
 <spring:message code="placeholder.user.password" var="placeHolderPassword"/>

 <!-- Navigation area -->
 <jsp:include page="navigation.jsp" />

 <div class="container">

  <!-- Breadcrumb area -->
  <ol class="breadcrumb">
   <li>
    <a href="/AgileReferential/home.html">
     <spring:message code="breadcrumb.home" />
    </a>
   </li>
   <li class="active">
    <spring:message code="breadcrumb.login" />
   </li>
  </ol>

  <!-- Alert area -->
```

```
  <jsp:include page="alert.jsp" />

  <!-- If login failed. Then notify the user. -->
  <c:if test="${param.error != null}">

   <div class="alert alert-danger" role="alert">
    <span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>
    <spring:message code="alert.message.login.error" />
   </div>

  </c:if>

  <!-- If the user logged out. Then notify him the operation succeeds -->
  <c:if test="${param.logout != null}">

   <div class="alert alert-info" role="alert">
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
     <span aria-hidden="true">&times;</span>
    </button>
    <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
    <spring:message code="alert.message.logout.sucess" />
   </div>

  </c:if>

  <div class="panel panel-default">

   <div class="panel-heading">

    <h3 class="panel-title">
         <span class="glyphicon glyphicon-log-in"></span>
         <spring:message code="panel.title.login" />
    </h3>

   </div>

   <div class="panel-body">

    <br />

    <div class="jumbotron col-sm-offset-2 col-sm-8">

     <!-- Login form -->
     <form:form name="login_form" role="form" class="form-signin" action="j_spring_security_check"
method="post">

      <!-- Login form - input field - email address -->
      <div class="form-group">
       <input type='text' class="form-control" id="inputEmail" name='email'
placeholder="${placeHolderEmail}" maxlength="128" autofocus />
      </div>

      <!-- Login form - input field - user password -->
      <div class="form-group">
       <input type='password' class="form-control" id="inputPassword" name='password'
       placeholder="${placeHolderPassword}" maxlength="64" />
      </div>
```

```
        <!-- Login form - submit button -->
        <button value="Login" class="btn btn-primary btn-block" type="submit">
          <spring:message code="button.login" />
        </button>
       </form:form>

      </div>

     </div>

   </div>

 </div>

</body>

</html>
```

8.4 listMethods.jsp

```
<!DOCTYPE HTML>

<html>

<head>

 <title>
  <spring:message code="title.method.list" />
 </title>

 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>
 <script src="js/list.js"></script>

</head>

<body>

 <spring:message code="placeholder.search" var="placeHolderSearch" />

 <!-- Navigation area -->
 <jsp:include page="navigation.jsp" />

 <div class="container">

  <!-- Breadcrumb area -->
  <ol class="breadcrumb">
   <li>
    <a href="/AgileReferential/home.html">
     <spring:message code="breadcrumb.home" />
    </a>
   </li>
```

```
  <li class="active">
    <spring:message code="breadcrumb.method.list" /> </li>
</ol>

<!-- Alert area -->
<jsp:include page="alert.jsp" />

<!-- Notify the user if the list of methods is empty. -->
<c:if test="${empty methodList}">
  <div class="alert alert-info" role="alert">
    <span class="glyphicon glyphicon-info-sign"></span>
    <spring:message code="alert.message.method.list.is.empty" />
  </div>
</c:if>

<div class="panel panel-default" id="filter_panorama">

  <div class="panel-heading">

    <div class="row">

      <div class="col-sm-6">
        <h3 class="panel-title">
          <span class="glyphicon glyphicon-list-alt"></span>
          <spring:message code="panel.title.method.list"/>
        </h3>
        <br/>
      </div>

      <div class="col-sm-6">
        <!-- Show search area if the list of methods is not empty. -->
        <c:if test="${!empty methodList}">
          <div class="input-group">
            <input type="text" class="form-control search" placeholder="${placeHolderSearch}"
            maxlength="128" autofocus>
            <span class="input-group-addon"><span class="glyphicon glyphiconsearch"></span>
            </span>
          </div>
        </c:if>
      </div>
    </div>

  </div>

  <div class="panel-body">

    <!-- Show add method button if the user has enough rights. -->
    <security:authorize access="hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')">
      <a href="<c:url value='/secure/addMethod'/>" class="btn large btn-primary btn-block">
        <spring:message code="button.add" />
      </a>
      <br/>
    </security:authorize>

    <div class="list">

      <!-- Show a method item for each method in the list -->
```

```
        <c:forEach items="${methodList}" var="method">
          <a href="<c:url value='method/details/${method.id}'/>" class="list-group-item">
            <span class="filter-name-criteria">${method.name}</span>
            <span class="glyphicon glyphicon-chevron-right pull-right"></span>
          </a>
        </c:forEach>

      </div>

    </div>

  </div>

 </div>

 <script type="text/javascript" src="<c:url value='/js/filtering.js'/>"></script>
</body>

</html>
```

8.5 listPractices.jsp

```
<%@ page   contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="security" uri="http://www.springframework.org/security/tags" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!DOCTYPE HTML>
<html>

<head>

 <title>
  <spring:message code="title.practice.list" />
 </title>

 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>
 <script src="js/list.js"></script>

</head>

<body>

 <spring:message code="placeholder.search" var="placeHolderSearch" />

 <!-- Navigation area -->
 <jsp:include page="navigation.jsp" />

 <div class="container">

  <!-- Breadcrumb area -->
  <ol class="breadcrumb">
```

```
  <li>
   <a href="/AgileReferential/home.html">
    <spring:message code="breadcrumb.home" />
   </a>
  </li>
  <li class="active">
   <spring:message code="breadcrumb.practice.list" /> </li>
</ol>

<!-- Alert area -->
<jsp:include page="alert.jsp" />

<!-- Notify the user if the list of practices is empty. -->
<c:if test="${empty practiceList}">
 <div class="alert alert-info" role="alert">
  <span class="glyphicon glyphicon-info-sign"></span>
  <spring:message code="alert.message.practice.list.is.empty" />
 </div>
</c:if>

<div class="panel panel-default" id="filter_panorama">

 <div class="panel-heading">

  <div class="row">

   <div class="col-sm-6">
    <h3 class="panel-title">
      <span class="glyphicon glyphicon-list-alt"></span>
      <spring:message code="panel.title.practice.list"/>
    </h3>
    <br/>
   </div>

   <div class="col-sm-6">
    <!-- Show search area if the list of practices is not empty. -->
    <c:if test="${!empty practiceList}">
     <div class="input-group">
      <input type="text" class="form-control search" placeholder="${placeHolderSearch}"
      maxlength="128" autofocus>
      <span class="input-group-addon">
              <span class="glyphicon glyphicon-search"></span>
      </span>
     </div>
    </c:if>

   </div>
  </div>

 </div>

 <div class="panel-body">

  <!-- Show add practice button if the user has enough rights. -->
  <security:authorize access="hasAnyRole('ROLE_ADMINISTRATOR', 'ROLE_EDITOR')">
   <a href="<c:url value='/secure/addPractice'/>" class="btn large btn-primary btn-block">
    <spring:message code="button.add" />
```

```
        </a>
        <br/>
      </security:authorize>

      <div class="list">
        <!-- Show a practice item for each practice in the list -->
        <c:forEach items="${practiceList}" var="practice">
          <a href="<c:url value='/practice/details/${practice.id}'/>" class="list-group-item">
            <span class="filter-name-criteria">${practice.name}</span>
            <span class="glyphicon glyphicon-chevron-right pull-right"></span>
          </a>
        </c:forEach>
      </div>

    </div>

   </div>

  </div>

 <script type="text/javascript" src="<c:url value='/js/filtering.js'/>"></script>

</body>

</html>
```

8.6 addMethod.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8"  %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form"  uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!DOCTYPE html>
<html>

<head>
 <title>
   <spring:message code="title.method.add" />
 </title>

 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/font-awesome/4.0.3/css/font-
awesome.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>

</head>

<body>

 <spring:message code="placeholder.method.name" var="placeHolderName" />

 <!-- Navigation area -->
```

```
<jsp:include page="navigation.jsp" />

<div class="container">

 <!-- Breadcrumb area -->
 <ol class="breadcrumb">
  <li>
   <a href="/AgileReferential/home.html">
    <spring:message code="breadcrumb.home" /> </a>
  </li>
  <li>
   <a href="/AgileReferential/methods.html">
    <spring:message code="breadcrumb.method.list" /> </a>
  </li>
  <li class="active">
   <spring:message code="breadcrumb.method.add" /> </li>
 </ol>

 <!-- Form for adding a new file for documenting a method -->
 <form:form class="form-horizontal" role="form" method="post" action="addMethod.html"
 commandName="method" data-toggle="validator">

  <!-- Alert area -->
  <jsp:include page="alertError.jsp" />

  <div class="panel panel-default">

   <div class="panel-heading">

    <h3 class="panel-title">
        <span class="glyphicon glyphicon-edit"></span>
        <spring:message code="panel.title.method.add"/>
    </h3>

   </div>

   <div class="panel-body" id="panel_overview">

    <!-- Form - input field - name of the method -->
    <spring:bind path="name">
     <div class="form-group ${status.error ? 'has-error has-feedback' : ''}">
      <label class="control-label col-sm-2" for="nameInput">
       <spring:message code="label.name" /> *</label>
      <div class="col-sm-8">
       <form:input type="text" class="form-control" id="nameInput" path="name"
       placeholder="${placeHolderName}" maxlength="40"></form:input>
       <form:errors path="name" class="help-block"></form:errors>
      </div>
     </div>
    </spring:bind>

    <hr>

    <div class="row">
     <!-- Cancellation button - Go back to the list of methods -->
     <p class="col-xs-12 col-sm-offset-6 col-sm-2">
      <a href="<c:url value='/methods.html'/>" class="btn btn-default btn-block">
```

```
            <spring:message code="button.cancel" />
           </a>
          </p>

          <!-- Confirmation button - Submit the form for adding a new method file -->
          <div class="col-xs-12 col-sm-2">
           <button type="submit" class="btn btn-success btn-block">
            <spring:message code="button.confirm" />
           </button>
          </div>
         </div>

       </div>

     </div>

   </form:form>

 </div>

</body>

</html>
```

8.7 addPractice.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8"  %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!DOCTYPE html>

<html>

<head>
 <title>
  <spring:message code="title.practice.add" />
 </title>

 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/font-awesome/4.0.3/css/font-
awesome.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>

</head>

<body>

 <spring:message code="placeholder.practice.name" var="placeHolderName"
 />

 <!-- Navigation area -->
```

```
<jsp:include page="navigation.jsp" />

<div class="container">

 <!-- Breadcrumb area -->
 <ol class="breadcrumb">
  <li>
   <a href="/AgileReferential/home.html">
    <spring:message code="breadcrumb.home" /> </a>
  </li>
  <li>
   <a href="/AgileReferential/practices.html">
    <spring:message code="breadcrumb.practice.list" /> </a>
  </li>
  <li class="active">
   <spring:message code="breadcrumb.practice.add" /> </li>
 </ol>

 <form:form class="form-horizontal" role="form" method="post" action="addPractice.html"
 commandName="practice" data-toggle="validator">

  <!-- Alert area -->
  <jsp:include page="alertError.jsp" />

  <div class="panel panel-default">

   <div class="panel-heading">

    <h3 class="panel-title">
        <span class="glyphicon glyphicon-edit"></span>
        <spring:message code="panel.title.practice.add"/>
    </h3>

   </div>

   <div class="panel-body" id="panel_overview">

    <!-- Form - input field - name of the practice -->
    <spring:bind path="name">
     <div class="form-group ${status.error ? 'has-error has-feedback' : ''}">
      <label class="control-label col-sm-2" for="nameInput">
       <spring:message code="label.name" /> *</label>
      <div class="col-sm-8">
       <form:input type="text" class="form-control" id="nameInput" path="name"
       placeholder="${placeHolderName}" maxlength="40"></form:input>
       <form:errors path="name" class="help-block"></form:errors>
      </div>
     </div>
    </spring:bind>

    <hr>

    <div class="row">

     <!-- Cancellation button - Go back to the list of practices -->
     <p class="col-xs-12 col-sm-offset-6 col-sm-2">
      <a href="<c:url value='/practices.html'/>" class="btn btn-default btn-block">
```

```
            <spring:message code="button.cancel" />
          </a>
         </p>

         <!-- Confirmation button - Submit the form for adding a new practice file -->
         <div class="col-xs-12 col-sm-2">
          <button type="submit" class="btn btn-success btn-block">
           <spring:message code="button.confirm" />
          </button>
         </div>
        </div>

      </div>

     </div>

   </form:form>

 </div>

</body>

</html>
```

8.8 addUser.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!DOCTYPE html>

<html>

<head>

 <title>
  <spring:message code="title.subscribe" />
 </title>

 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
 <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/font-awesome/4.0.3/css/font-
awesome.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
 <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>

</head>

<body>

 <spring:message code="placeholder.user.email" var="placeHolderEmail" />
 <spring:message code="placeholder.user.firstname" var="placeHolderFirstName"/>
 <spring:message code="placeholder.user.lastname" var="placeHolderLastName"/>
```

```
<spring:message code="placeholder.user.password" var="placeHolderPassword"/>

<!-- Navigation area -->
<jsp:include page="navigation.jsp" />

<div class="container">

 <!-- Breadcrumb area -->
 <ol class="breadcrumb">
  <li>
   <a href="/AgileReferential/home.html">
    <spring:message code="breadcrumb.home" />
   </a>
  </li>
  <li class="active">
   <spring:message code="breadcrumb.subscribe" />
  </li>
 </ol>

 <!-- Subscription form -->
 <form:form class="form-horizontal" role="form" method="post" action="subscribe.html"
 commandName="user" data-toggle="validator">

  <!-- Alert area -->
  <jsp:include page="alertError.jsp" />

  <div class="panel panel-default">

   <div class="panel-heading">

    <h3 class="panel-title">
        <span class="glyphicon glyphicon-user"></span>
        <spring:message code="panel.title.subscribe"/>
    </h3>

   </div>

   <div class="panel-body" id="panel_overview">

    <!-- Form - input field - email address -->
    <spring:bind path="email">
     <div class="form-group ${status.error ? 'has-error has-feedback' : ''}">
      <label class="control-label col-sm-2" for="emailInput">
       <spring:message code="label.email" /> *</label>
      <div class="col-sm-8">
       <form:input type="text" class="form-control" id="emailInput" path="email"
       placeholder="${placeHolderEmail}" maxlength="128"></form:input>
       <form:errors path="email" class="help-block"></form:errors>
      </div>
     </div>
    </spring:bind>

    <!-- Form - input field - user password -->
    <spring:bind path="password">
     <div class="form-group ${status.error ? 'has-error' : ''}">
      <label class="control-label col-sm-2" for="passwordInput">
       <spring:message code="label.password" /> *</label>
```

```
          <div class="col-sm-8">
            <form:input type="password" class="form-control" id="passwordInput" path="password"
            placeholder="${placeHolderPassword}" maxlength="64"></form:input>
            <form:errors path="password" class="help-block"></form:errors>
          </div>
        </div>
      </spring:bind>

      <!-- Form - input field - user last name -->
      <spring:bind path="lastName">
        <div class="form-group ${status.error ? 'has-error has-feedback' : ''}">
          <label class="control-label col-sm-2" for="lastNameInput">
            <spring:message code="label.lastname" /> *</label>
          <div class="col-sm-8">
            <form:input type="text" class="form-control" id="lastNameInput" path="lastName"
            placeholder="${placeHolderLastName}" maxlength="64"></form:input>
            <form:errors path="lastName" class="help-block"></form:errors>
          </div>
        </div>
      </spring:bind>

      <!-- Form - input field - user firs tname -->
      <spring:bind path="firstName">
        <div class="form-group ${status.error ? 'has-error has-feedback' : ''}">
          <label class="control-label col-sm-2" for="firstNameInput">
            <spring:message code="label.firstname" /> *</label>
          <div class="col-sm-8">
            <form:input type="text" class="form-control" id="firstNameInput" path="firstName"
            placeholder="${placeHolderFirstName}" maxlength="64"></form:input>
            <form:errors path="firstName" class="help-block"></form:errors>
          </div>
        </div>
      </spring:bind>

      <hr>

      <!-- Form - submit button -->
      <div class="row">
        <div class="col-xs-12 col-sm-offset-8 col-sm-2">
          <button type="submit" class="btn btn-success btn-block">
            <spring:message code="button.confirm" />
          </button>
        </div>
      </div>

    </div>

  </div>

 </form:form>

</div>

</body>

</html>
```

8.9 modalDeleteMethod.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="spring"   uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="confirm-delete-method" tabindex="-1" role="dialog">
 <div class="modal-dialog">
  <div class="modal-content">

    <div class="modal-header">
     <button type="button" class="close" data-dismiss="modal">&times;</button>
     <h4 class="modal-title">
          <span class="glyphicon glyphicon-exclamation-sign"></span>
          <spring:message code="label.confirmation"/>
     </h4>
    </div>

    <div class="modal-body">
     <p>
      <spring:message code="text.modal.confirm.delete.method" />
     </p>
    </div>

    <div class="modal-footer">
     <button type="button" class="btn btn-default" data-dismiss="modal">
      <spring:message code="button.cancel" />
     </button>
     <a href="/AgileReferential/secure/deleteMethod/${method.id}" class="btn btn-danger">
      <spring:message code="button.delete" />
     </a>
    </div>

  </div>
 </div>
</div>
```

8.10 modalDeletePractice.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="confirm-delete-practice" tabindex="-1" role="dialog">
 <div class="modal-dialog">
  <div class="modal-content">

    <div class="modal-header">
     <button type="button" class="close" data-dismiss="modal">&times;</button>
     <h4 class="modal-title">
          <span class="glyphicon glyphicon-exclamation-sign"></span>
          <spring:message code="label.confirmation"/>
     </h4>
    </div>

    <div class="modal-body">
     <p>
      <spring:message code="text.modal.confirm.delete.practice" />
     </p>
    </div>
```

```
    <div class="modal-footer">
     <button type="button" class="btn btn-default" data-dismiss="modal">
      <spring:message code="button.cancel" />
     </button>
     <a href="/AgileReferential/secure/deletePractice/${practice.id}" class="btn btn-danger">
      <spring:message code="button.delete" />
     </a>
    </div>

   </div>
  </div>
</div>
```

8.11 modalFormEditMethodDescription.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_method_description" tabindex="-1"
role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.method.edit.description"/>
    </h4>
   </div>

   <!-- Form to edit the description of a method -->
   <form:form class="form-horizontal" role="form" method="post" action="editMethod.html"
   commandName="method" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.practice.description" var="placeHolderDescription" />

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
     <form:input path="scope" id="scopeInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="adoptionRecommendations" id="adoptionRecommendationsInput"
     type="hidden"></form:input>

     <div class="hiddenDiv">
      <form:checkboxes path="practices" element="p class='checkBoxStyle filter_on_name'"
      items="${existingPracticeList}" itemLabel="name" itemValue="id"/>
     </div>

     <form:textarea class="form-control summernote" path="description"
placeholder="${placeHolderDescription}" rows="10"></form:textarea>
    </div>

    <div class="modal-footer fullscreen">
     <div class="row">
```

```
        <p class="col-xs-6 col-md-offset-8 col-md-2">
          <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
            <spring:message code="button.cancel" />
          </button>
        </p>
        <p class="col-xs-6 col-md-2">
          <button type="submit" class="btn btn-success btn-sm btn-block">
            <spring:message code="button.confirm" />
          </button>
        </p>
       </div>
      </div>

    </form:form>

   </div>
  </div>
</div>
```

8.12 modalFormEditMethodScope.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_method_scope" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.method.edit.scope"/>
    </h4>
   </div>

   <!-- Form to edit the scope of a method -->
   <form:form class="form-horizontal" role="form" method="post" action="editMethod.html"
   commandName="method" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.method.scope" var="placeHolderScope"/>

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
     <form:input path="description" id="descriptionInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="adoptionRecommendations" id="adoptionRecommendationsInput"
     type="hidden"></form:input>

     <div class="hiddenDiv">
      <form:checkboxes path="practices" element="p class='checkBoxStyle filter_on_name'"
      items="${existingPracticeList}" itemLabel="name" itemValue="id"/>
     </div>

     <form:textarea class="form-control summernote" path="scope" placeholder="${placeHolderScope}"
```

```
      rows="10"></form:textarea>
     </div>

     <div class="modal-footer fullscreen">
      <div class="row">
       <p class="col-xs-6 col-md-offset-8 col-md-2">
        <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
         <spring:message code="button.cancel" />
        </button>
       </p>
       <p class="col-xs-6 col-md-2">
        <button type="submit" class="btn btn-success btn-sm btn-block">
         <spring:message code="button.confirm" />
        </button>
       </p>
      </div>
     </div>

    </form:form>

   </div>
  </div>
</div>
```

8.13 modalFormEditMethodAdoptionRecommendations.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_method_adoption_recommendations"
tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.method.edit.adoption.recommendations"/>
    </h4>
   </div>

   <!-- Form to edit the adoption recommendations about a method -->
   <form:form class="form-horizontal" role="form" method="post" action="editMethod.html"
   commandName="method" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.adoption.recommendations"
var="placeHolderAdoptionRecommendations"/>

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="description" id="descriptionInput" type="hidden"></form:input>
     <form:input path="scope" id="scopeInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
```

```
        <div class="hiddenDiv">
          <form:checkboxes path="practices" element="p class='checkBoxStyle filter_on_name'"
          items="${existingPracticeList}" itemLabel="name" itemValue="id"/>
        </div>

        <form:textarea class="form-control summernote" path="adoptionRecommendations"
        placeholder="${placeHolderAdoptionRecommendations}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
        <div class="row">
          <p class="col-xs-6 col-md-offset-8 col-md-2">
            <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
              <spring:message code="button.cancel" />
            </button>
          </p>
          <p class="col-xs-6 col-md-2">
            <button type="submit" class="btn btn-success btn-sm btn-block">
              <spring:message code="button.confirm" />
            </button>
          </p>
        </div>
      </div>

    </form:form>

  </div>
 </div>
</div>
```

8.14 modalFormEditMethodOrigins.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_method_origins" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

    <div class="modal-header fullscreen">
      <button type="button" class="close" data-dismiss="modal">&times;</button>
      <h4 class="modal-title fullscreen">
            <span class="glyphicon glyphicon-edit"></span>
            <spring:message code="modal.title.method.edit.origins"/>
      </h4>
    </div>

    <!-- Form to edit the historical context of a method -->
    <form:form class="form-horizontal" role="form" method="post" action="editMethod.html"
    commandName="method" data-toggle="validator">

      <div class="modal-body fullscreen">
        <spring:message code="placeholder.practice.origins" var="placeHolderOrigins"/>

        <form:input path="id" id="idInput" type="hidden"></form:input>
        <form:input path="name" id="nameInput" type="hidden"></form:input>
        <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
```

```
        <form:input path="scope" id="scopeInput" type="hidden"></form:input>
        <form:input path="description" id="descriptionInput" type="hidden"></form:input>
        <form:input path="adoptionRecommendations" id="adoptionRecommendationsInput"
        type="hidden"></form:input>

        <div class="hiddenDiv">
          <form:checkboxes path="practices" element="p class='checkBoxStyle filter_on_name'"
          items="${existingPracticeList}" itemLabel="name" itemValue="id"/>
        </div>

        <form:textarea class="form-control summernote" path="historicalContext"
        placeholder="${placeHolderOrigins}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
        <div class="row">
          <p class="col-xs-6 col-md-offset-8 col-md-2">
            <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
              <spring:message code="button.cancel" />
            </button>
          </p>
          <p class="col-xs-6 col-md-2">
            <button type="submit" class="btn btn-success btn-sm btn-block">
              <spring:message code="button.confirm" />
            </button>
          </p>
        </div>
      </div>

    </form:form>

  </div>
 </div>
</div>
```

8.15 modalFormEditMethodBibliography.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<div class="modal fade" id="form_edit_method_bibliography" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

    <div class="modal-header fullscreen">
      <button type="button" class="close" data-dismiss="modal">&times;</button>
      <h4 class="modal-title fullscreen">
            <span class="glyphicon glyphicon-edit"></span>
            <spring:message code="modal.title.method.edit.bibliography"/>
      </h4>
    </div>

    <!-- Form to edit the bibliography of a method -->
    <form:form class="form-horizontal" role="form" method="post" action="editMethod.html"
    commandName="method" data-toggle="validator">

      <div class="modal-body fullscreen">
        <spring:message code="placeholder.bibliography" var="placeHolderBibliography"/>
```

```
        <form:input path="id" id="idInput" type="hidden"></form:input>
        <form:input path="name" id="nameInput" type="hidden"></form:input>
        <form:input path="description" id="descriptionInput" type="hidden"></form:input>
        <form:input path="scope" id="scopeInput" type="hidden"></form:input>
        <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
        <form:input path="adoptionRecommendations" id="adoptionRecommendationsInput" type="hidden">
        </form:input>

        <div class="hiddenDiv">
          <form:checkboxes path="practices" element="p class='checkBoxStyle filter_on_name'"
          items="${existingPracticeList}" itemLabel="name" itemValue="id"/>
        </div>

        <form:textarea class="form-control summernote" path="bibliography"
placeholder="${placeHolderBibliography}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
        <div class="row">
          <p class="col-xs-6 col-md-offset-8 col-md-2">
            <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
              <spring:message code="button.cancel" />
            </button>
          </p>
          <p class="col-xs-6 col-md-2">
            <button type="submit" class="btn btn-success btn-sm btn-block">
              <spring:message code="button.confirm" />
            </button>
          </p>
        </div>
      </div>

    </form:form>

   </div>
  </div>
</div>
```

8.16 modalFormEditMethodPractices.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_method_practices" tabindex="-1" role="dialog">

  <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
    <div class="modal-content fullscreen">

      <div class="modal-header fullscreen">
        <button type="button" class="close" data-dismiss="modal">&times;</button>
        <h4 class="modal-title fullscreen">
            <span class="glyphicon glyphicon-edit"></span>
            <spring:message code="modal.title.method.link.practices"/>
        </h4>
      </div>
```

```
<div id="filter_link_practices">

 <!-- Form to link/unlink practices to a method -->
 <form:form id="form_link_practices_to_method" class="form-horizontal" role="form"
 method="post" action="editMethod.html" commandName="method" data-toggle="validator">

  <spring:message code="placeholder.search" var="placeHolderSearch" />

  <div class="modal-body fullscreen">

   <form:input path="id" type="hidden"></form:input>
   <form:input path="name" type="hidden"></form:input>
   <form:input path="description" type="hidden"></form:input>
   <form:input path="bibliography" type="hidden"></form:input>
   <form:input path="scope" type="hidden"></form:input>
   <form:input path="historicalContext" type="hidden"></form:input>
   <form:input path="adoptionRecommendations" type="hidden"></form:input>

   <!-- Filtering on method - Dropdown list -->
   <div class="col-xs-12 col-lg-6 noPadding">
    <div class="form-group">
     <select id="filter_methods" class="filter form-control">
      <optgroup label="<spring:message code=" select.filter.practices.optgroup"/>">
       <option value="*">
        <spring:message code="select.filter.practices.opt.all"/>
       </option>
       <c:forEach varStatus="status" var="existingMethod" items="${existingMethodList}">
        <option value="${existingMethod.id}">
         <spring:message code="select.filter.practices.prefix"/> ${existingMethod.name}</option>
       </c:forEach>
      </optgroup>
     </select>
    </div>
   </div>

   <div class="col-lg-2"></div>

   <div class="col-xs-12 col-lg-4 noPadding">
    <!-- Free filtering - Input field -->
    <div class="form-group">
     <div class="input-group">
      <input type="text" class="form-control search" placeholder="${placeHolderSearch}"
      maxlength="40" autofocus>
      <span class="input-group-addon"><span class="glyphicon glyphicon-search"></span></span>
     </div>
    </div>
   </div>

   <c:if test="${!empty existingPracticeList}">

    <!-- Filtered list of practices -->
    <div class="form-group checkBoxStyle">
     <ul>
      <li>
       <hr>
       <input type="checkbox" id="selectall">
```

```
                    <b><spring:message code="select.filter.practices.check.all"/></b>
                </input>
                <hr>
              </li>
            </ul>
            <ul class="list">

              <c:forEach varStatus="status" var="practice" items="${existingPracticeList}">
                <li>

                  <c:set var="checked" value="" scope="page" />

                  <c:forEach varStatus="status" var="metho" items="${method.practices}">
                    <c:if test="${metho.id == practice.id}">
                      <c:set var="checked" value="1" />
                    </c:if>
                  </c:forEach>

                  <c:choose>
                    <c:when test="${!empty checked}">
                      <form:checkbox path="practices" value="${practice.id}" checked="checked"/>
                    </c:when>
                    <c:otherwise>
                      <form:checkbox path="practices" value="${practice.id}" />
                    </c:otherwise>
                  </c:choose>

                  <span class="filter_practice_name_criteria">${practice.name}</span>
                  <div class="hiddenDiv">
                    <span class="filter_method_name_criteria">
                        <c:forEach varStatus="status" var="linkedMethod" items="${practice.methods}">
                        ;${linkedMethod.id};
                        </c:forEach>
                    </span>
                  </div>

                  <hr>

                </li>
              </c:forEach>
            </ul>
          </div>
        </c:if>
      </div>

      <div class="modal-footer fullscreen">
        <div class="row">
          <p class="col-xs-6 col-md-offset-8 col-md-2">
            <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
              <spring:message code="button.cancel" />
            </button>
          </p>
          <p class="col-xs-6 col-md-2">
            <button type="submit" class="btn btn-success btn-sm btn-block">
              <spring:message code="button.confirm" />
            </button>
          </p>
```

```
        </div>
       </div>

      </form:form>
     </div>
    </div>
   </div>
</div>
```

8.17 modalFormEditPracticeDescription.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_description" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
        <span class="glyphicon glyphicon-edit"></span>
        <spring:message code="modal.title.practice.edit.description"/>
    </h4>
   </div>

   <!-- Form to edit the description of a practice -->
   <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
   commandName="practice" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.practice.description" var="placeHolderDescription"/>

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
     <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
     <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
     <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
     <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
     </form:input>

     <form:textarea class="form-control summernote" path="description"
placeholder="${placeHolderDescription}" rows="10"></form:textarea>
    </div>

    <div class="modal-footer fullscreen">
     <div class="row">
      <p class="col-xs-6 col-md-offset-8 col-md-2">
       <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
        <spring:message code="button.cancel" />
       </button>
      </p>
      <p class="col-xs-6 col-md-2">
```

```
          <button type="submit" class="btn btn-success btn-sm btn-block">
            <spring:message code="button.confirm" />
          </button>
        </p>
      </div>
    </div>

  </form:form>

 </div>
</div>
</div>
```

8.18 modalFormEditPracticeBenefits.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_benefits" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.practice.edit.benefits"/>
    </h4>
   </div>

   <!-- Form to edit the benefits of a practice -->
   <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
   commandName="practice" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.practice.benefits" var="placeHolderBenefits"/>

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="description" id="descriptionInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
     <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
     <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
     <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
     <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
     </form:input>

     <form:textarea class="form-control summernote" path="benefits" placeholder="${placeHolderBenefits}"
     rows="10"></form:textarea>
    </div>

    <div class="modal-footer fullscreen">
     <div class="row">
      <p class="col-xs-6 col-md-offset-8 col-md-2">
       <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
```

```
            <spring:message code="button.cancel" />
           </button>
          </p>
          <p class="col-xs-6 col-md-2">
           <button type="submit" class="btn btn-success btn-sm btn-block">
            <spring:message code="button.confirm" />
           </button>
          </p>
         </div>
        </div>

    </form:form>

   </div>
  </div>
</div>
```

8.19 modalFormEditPracticeOrigins.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_origins" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.practice.edit.origins"/></h4>
   </div>

   <!-- Form to edit the origins of a practice -->
   <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
   commandName="practice" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.practice.origins" var="placeHolderOrigins" />

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
     <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
     <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
     <form:input path="description" id="descriptionInput" type="hidden"></form:input>
     <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
     <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
     <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
     </form:input>

     <form:textarea class="form-control summernote" path="historicalContext"
     placeholder="${placeHolderOrigins}" rows="10"></form:textarea>
    </div>

    <div class="modal-footer fullscreen">
```

```
        <div class="row">
         <p class="col-xs-6 col-md-offset-8 col-md-2">
          <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
           <spring:message code="button.cancel" />
          </button>
         </p>
         <p class="col-xs-6 col-md-2">
          <button type="submit" class="btn btn-success btn-sm btn-block">
           <spring:message code="button.confirm" />
          </button>
         </p>
        </div>
       </div>

     </form:form>

    </div>
   </div>
</div>
```

8.20 modalFormEditPracticeSynonyms.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_synonyms" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.practice.edit.synonyms"/>
    </h4>
   </div>

   <!-- Form to edit the synonyms of a practice -->
   <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
   commandName="practice" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.practice.synonyms" var="placeHolderSynonyms"/>

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
     <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
     <form:input path="description" id="descriptionInput" type="hidden"></form:input>
     <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
     <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
     <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
     </form:input>

     <form:textarea class="form-control summernote" path="synonyms"
```

```
placeholder="${placeHolderSynonyms}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
       <div class="row">
        <p class="col-xs-6 col-md-offset-8 col-md-2">
         <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
          <spring:message code="button.cancel" />
         </button>
        </p>
        <p class="col-xs-6 col-md-2">
         <button type="submit" class="btn btn-success btn-sm btn-block">
          <spring:message code="button.confirm" />
         </button>
        </p>
       </div>
      </div>

     </form:form>

   </div>
  </div>
</div>
```

8.21 modalFormEditPracticeBibliography.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_bibliography" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.practice.edit.bibliography"/>
    </h4>
   </div>

   <!-- Form to edit the bibliography of a practice -->
   <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
   commandName="practice" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.bibliography" var="placeHolderBibliography"/>

     <form:input path="id" id="idInput" type="hidden"></form:input>
     <form:input path="name" id="nameInput" type="hidden"></form:input>
     <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
     <form:input path="description" id="descriptionInput" type="hidden"></form:input>
     <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
     <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
     <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
     <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
     <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
```

```
        <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
        </form:input>

        <form:textarea class="form-control summernote" path="bibliography"
placeholder="${placeHolderBibliography}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
        <div class="row">
          <p class="col-xs-6 col-md-offset-8 col-md-2">
            <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
              <spring:message code="button.cancel" />
            </button>
          </p>
          <p class="col-xs-6 col-md-2">
            <button type="submit" class="btn btn-success btn-sm btn-block">
              <spring:message code="button.confirm" />
            </button>
          </p>
        </div>
      </div>

    </form:form>

   </div>
  </div>
</div>
```

8.22 modalFormEditPracticeCommonMistakes.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_common_mistakes" tabindex="-1" role="dialog">
  <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
    <div class="modal-content fullscreen">

      <div class="modal-header fullscreen">
        <button type="button" class="close" data-dismiss="modal">&times;</button>
        <h4 class="modal-title fullscreen">
            <span class="glyphicon glyphicon-edit"></span>
            <spring:message code="modal.title.practice.edit.common.mistakes"/>
        </h4>
      </div>

      <!-- Form to edit the common mistakes related to the usage of a practice -->
      <form:form id="form_edit_common_mistakes" class="form-horizontal" role="form"
      method="post" action="editPractice.html" commandName="practice" data-toggle="validator">

        <div class="modal-body fullscreen">
          <spring:message code="placeholder.practice.common.mistakes" var="placeHolderCommonMistakes"/>

          <form:input path="id" id="idInput" type="hidden"></form:input>
          <form:input path="name" id="nameInput" type="hidden"></form:input>
          <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
          <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
          <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
```

```
      <form:input path="description" id="descriptionInput" type="hidden"></form:input>
      <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
      <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
      <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
      <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
      </form:input>

      <form:textarea class="form-control summernote" path="commonMistakes" id="commonMistakesInput"
      placeholder="${placeHolderCommonMistakes}" rows="10"></form:textarea>
     </div>

     <div class="modal-footer fullscreen">
      <div class="row">
       <p class="col-xs-6 col-md-offset-8 col-md-2">
        <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
         <spring:message code="button.cancel" />
        </button>
       </p>
       <p class="col-xs-6 col-md-2">
        <button type="submit" class="btn btn-success btn-sm btn-block">
         <spring:message code="button.confirm" />
        </button>
       </p>
      </div>
     </div>

    </form:form>

   </div>
  </div>
</div>
```

8.23 modalFormEditPracticeObservableSigns.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_observable_signs" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
         <span class="glyphicon glyphicon-edit"></span>
         <spring:message code="modal.title.practice.edit.observable.signs"/>
    </h4>
   </div>

   <!-- Form to edit the observable signs of a practice -->
   <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
   commandName="practice" data-toggle="validator">

    <div class="modal-body fullscreen">
     <spring:message code="placeholder.practice.observable.signs" var="placeHolderObservableSigns"/>

      <form:input path="id" id="idInput" type="hidden"></form:input>
```

```
        <form:input path="name" id="nameInput" type="hidden"></form:input>
        <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
        <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
        <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
        <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
        <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
        <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
        <form:input path="description" id="descriptionInput" type="hidden"></form:input>
        <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
        </form:input>

        <form:textarea class="form-control summernote" path="observableSigns"
placeholder="${placeHolderObservableSigns}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
        <div class="row">
          <p class="col-xs-6 col-md-offset-8 col-md-2">
            <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
              <spring:message code="button.cancel" />
            </button>
          </p>
          <p class="col-xs-6 col-md-2">
            <button type="submit" class="btn btn-success btn-sm btn-block">
              <spring:message code="button.confirm" />
            </button>
          </p>
        </div>
      </div>

    </form:form>

  </div>
 </div>
</div>
```

8.24 modalFormEditPracticeImprovementTargets.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_improvement_targets" tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

    <div class="modal-header fullscreen">
     <button type="button" class="close" data-dismiss="modal">&times;</button>
     <h4 class="modal-title fullscreen">
          <span class="glyphicon glyphicon-edit"></span>
          <spring:message code="modal.title.practice.edit.improvement.targets"/>
      </h4>
    </div>

    <!-- Form to edit the improvement targerts of a practice -->
    <form:form id="form_edit_improvement_targets" class="form-horizontal" role="form"
    method="post" action="editPractice.html" commandName="practice" data-toggle="validator">
```

```
      <div class="modal-body fullscreen">
       <spring:message code="placeholder.practice.improvement.targets"
var="placeHolderImprovementTargets"/>

       <form:input path="id" id="idInput" type="hidden"></form:input>
       <form:input path="name" id="nameInput" type="hidden"></form:input>
       <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
       <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
       <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
       <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
       <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
       <form:input path="description" id="descriptionInput" type="hidden"></form:input>
       <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
       <form:input path="adoptionRecommendations" id="recommendationsInput" type="hidden">
       </form:input>

       <form:textarea class="form-control summernote" path="improvementTargets"
       id="improvementTargetsInput" placeholder="${placeHolderImprovementTargets}"
       rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
       <div class="row">
        <p class="col-xs-6 col-md-offset-8 col-md-2">
         <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
          <spring:message code="button.cancel" />
         </button>
        </p>

        <p class="col-xs-6 col-md-2">
         <button type="submit" class="btn btn-success btn-sm btn-block">
          <spring:message code="button.confirm" />
         </button>
        </p>
       </div>
      </div>

     </form:form>

    </div>
   </div>
</div>
```

8.25 modalFormEditPracticeAdoptionRecommendations.jsp

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<div class="modal fade" id="form_edit_practice_adoption_recommendations"
tabindex="-1" role="dialog">
 <div class="modal-dialog fullscreen modal-fullscreen force-fullscreen">
  <div class="modal-content fullscreen">

   <div class="modal-header fullscreen">
    <button type="button" class="close" data-dismiss="modal">&times;</button>
    <h4 class="modal-title fullscreen">
        <span class="glyphicon glyphicon-edit"></span>
```

```
          <spring:message code="modal.title.practice.edit.adoption.recommendations"/>
      </h4>
    </div>

    <!-- Form to edit the adoption recommendations over a practice -->
    <form:form class="form-horizontal" role="form" method="post" action="editPractice.html"
    commandName="practice" data-toggle="validator">

      <div class="modal-body fullscreen">
       <spring:message code="placeholder.adoption.recommendations"
var="placeHolderAdoptionRecommendations"/>

       <form:input path="id" id="idInput" type="hidden"></form:input>
       <form:input path="name" id="nameInput" type="hidden"></form:input>
       <form:input path="benefits" id="benefitsInput" type="hidden"></form:input>
       <form:input path="bibliography" id="bibliographyInput" type="hidden"></form:input>
       <form:input path="synonyms" id="synonymsInput" type="hidden"></form:input>
       <form:input path="commonMistakes" id="commonMistakesInput" type="hidden"></form:input>
       <form:input path="historicalContext" id="historicalContextInput" type="hidden"></form:input>
       <form:input path="improvementTargets" id="improvementTargetsInput" type="hidden"></form:input>
       <form:input path="observableSigns" id="observableSignsInput" type="hidden"></form:input>
       <form:input path="description" id="descriptionInput" type="hidden"></form:input>

       <form:textarea class="form-control summernote" path="adoptionRecommendations"
       placeholder="${placeHolderAdoptionRecommendations}" rows="10"></form:textarea>
      </div>

      <div class="modal-footer fullscreen">
       <div class="row">
        <p class="col-xs-6 col-md-offset-8 col-md-2">
         <button type="button" class="btn btn-default btn-sm btn-block" data-dismiss="modal">
          <spring:message code="button.cancel" />
         </button>
        </p>
        <p class="col-xs-6 col-md-2">
         <button type="submit" class="btn btn-success btn-sm btn-block">
          <spring:message code="button.confirm" />
         </button>
        </p>
       </div>
      </div>

    </form:form>

   </div>
  </div>
</div>
```

8.26 alert.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!-- If a success message has been posted. Then, show that message to the user. -->
<c:if test="${!empty alertSuccessMessage}">

          <div class="alert alert-success alert-dismissible" role="alert">
```

```
                <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                </button>
                <span class="glyphicon glyphicon-ok-sign" aria-hidden="true"></span>
                <spring:message code="${alertSuccessMessage}" />

        </div>

</c:if>
```

8.27 alertError.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"  %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<!-- In case of an error, bind the error and show the related error message. -->
<spring:bind path="*">
        <c:if test="${status.error}">

                <div class="alert alert-danger" role="alert">
                  <span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>
                  <spring:message code="alert.message.form.error" />
                </div>

        </c:if>
</spring:bind>
```

8.28 navigation.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="security" uri="http://www.springframework.org/security/tags" %>
<%@ taglib prefix="spring"   uri="http://www.springframework.org/tags" %>

<c:url value="/j_spring_security_logout" var="logoutUrl" />


<div class="navbar navbar-inverse navbar-static-top">

 <div class="navbar-header">

  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#navbar">
   <span class="icon-bar"></span>
   <span class="icon-bar"></span>
   <span class="icon-bar"></span>
  </button>

  <!-- Home link -->

  <a class="navbar-brand" href="/AgileReferential/home.html">
   <span class="glyphicon glyphicon-home"></span>
  </a>

 </div>

 <div id="navbar" class="collapse navbar-collapse">

  <!-- Links to main sections. -->

  <ul class="nav navbar-nav">
```

```
    <li>
     <!-- Link to the list of methods. -->
     <a href="/AgileReferential/methods.html">
      <spring:message code="navigation.methods" />
     </a>
    </li>
    <li>
     <!-- Link to the list of practices. -->
     <a href="/AgileReferential/practices.html">
      <spring:message code="navigation.practices" />
     </a>
    </li>
   </ul>

   <ul class="nav navbar-nav navbar-right">

    <!-- If the user is not authenticated. Then display login and subscription links. -->

    <security:authorize access="isAnonymous()">
     <li>
      <a href="<c:url value='/subscribe'/>">
       <span class="glyphicon glyphicon-user"></span>
       <spring:message code="navigation.subscribe" />
      </a>
     </li>
     <li>
      <a href="<c:url value='/login'/>">
       <span class="glyphicon glyphicon-log-in"></span>
       <spring:message code="navigation.login" />
      </a>
     </li>
     <li>      </li>
    </security:authorize>

    <!-- If the user is authenticated. Then display logout link. -->

    <security:authorize access="isAuthenticated()">
     <li>
      <a href="<c:url value=" ${logoutUrl} " />">
       <span class="glyphicon glyphicon-log-out"></span>
       <spring:message code="navigation.logout" />
      </a>
     </li>
     <li>      </li>
    </security:authorize>

   </ul>

  </div>

</div>
```

9 Fichiers CSS

9.1 reset.css

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6,
p, blockquote, pre, a, abbr, acronym, address, big, cite, code, del,
dfn, em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup,
tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label,
legend, table, caption, tbody, tfoot, thead, tr, th, td, article,
aside, canvas, details, embed, figure, figcaption, footer, header,
hgroup, menu, nav, output, ruby, section, summary, time, mark, audio,
video {
  margin: 0;
  padding: 0;
  border: 0;
  font - size: 100 % ;
  font: inherit;
  vertical - align: baseline;
}

/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure, footer, header, hgroup,
menu, nav, section {
  display: block;
}

body {
  line - height: 1;
}

ol, ul {
  list - style: none;
}

blockquote, q {
  quotes: none;
}

blockquote: before, blockquote: after, q: before, q: after {
  content: '';
  content: none;
}

table {
  border - collapse: collapse;
  border - spacing: 0;
}
```

9.2 main.css

```
html, body, .container {
  height: 100 % ;
}

.row - fluid {
  word - wrap: break -word;
}

p {
```

```
  word - wrap: break -word;
}

.noPadding {
  padding - left: 0 px;
  padding - right: 0 px;
}

.commentBox {
  background - color: white;
  border: 1 px solid# ddd;
  margin: 5 px;
  padding: 10 px;
}

textarea {
  min - height: 100 % ;
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  resize: none;
}

.modal - dialog.fullscreen {
  position: fixed;
  margin: 0;
  width: 100 % ;
  height: 100 % ;
  padding: 0;
}

.modal - content.fullscreen {
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  border: 2 px solid;
  border - radius: 0;
  box - shadow: none;
}

.modal - header.fullscreen {
  position: absolute;
  top: 0;
  right: 0;
  left: 0;
  height: 50 px;
  padding: 10 px;
}

.modal - title.fullscreen {
  line - height: 30 px;
}
```

```
.modal - body.fullscreen {
  position: absolute;
  top: 50 px;
  bottom: 60 px;
  width: 96 % ;
  overflow: auto;
  right: 2 % ;
  left: 2 % ;
}

.modal - footer.fullscreen {
  position: absolute;
  right: 0;
  bottom: 0;
  left: 0;
  height: 60 px;
  padding: 10 px;
}

.modal - vertical - centered {
  transform: translate(0, 50 % ) !important; - ms - transform: translate(
    0, 50 % ) !important; /* IE 9 */ - webkit - transform: translate(
    0, 50 % ) !important;
  /* Safari and Chrome */
}

.icon - black {
  color: black;
}

.checkBoxStyle {
  list - style - type: none;
}

.checkBoxStyle ul {
  padding: 0;
  margin: 0;
  width: 100 % ;
  overflow: auto;
  right: 0;
  left: 0;
  list - style - type: none;
}

.checkBoxStyle input[type = checkbox] {
  margin: 4 px 5 px 0;
  float: left;
}

.hiddenDiv {
  display: none;
}

.backgroundGrey {
  background - color: grey;
}
```

10 Fichiers JavaScript

10.1 texteditor.js

```
// Initialize summernote texteditor
$(document).ready(function() {

 $(document).ready(function() {
  $('[data-toggle="tooltip"]').tooltip();
 });

 $('.summernote').summernote({

  height: 300,
  disableResizeEditor: true,
  lang: 'fr-FR',
  toolbar: [
   //[groupname, [button list]]

   ['style', ['bold', 'italic', 'underline', 'clear']],
   ['para', ['ul', 'ol']],
   ['history', ['undo']],
  ]
 });

 $('.note-editable').trigger('focus');

});

$('.modal').on('show.bs.modal', centerModal);
$(window).on("resize", function() {
 $('.modal:visible').each(centerModal);
});

(function($) {
 $.extend($.summernote.lang, {
  'fr-FR': {
   font: {
    bold: 'Gras',
    italic: 'Italique',
    underline: 'Souligné',
    clear: 'Effacer la mise en forme',
   },
   lists: {
    unordered: 'Liste à puces',
    ordered: 'Liste numérotée'
   },
   history: {
    undo: 'Annuler la dernière action',
   }
  }
 });
})(jQuery);
```

10.2 filtering.js

```
/*****************************************
 * Filter for methods & practices panorama        *
 *****************************************/
```

```
// Set filters criteria
var filterPanoramaOptions = {valueNames: ['filter-name-criteria']
};

// Initialize filter
var filterPanorama = new List('filter_panorama', filterPanoramaOptions);

/*********************************************
 * Filter for the screen - Link/Unlink practices                    *
 *********************************************/

// Set filters criteria
var filterLinkPracticesOptions={valueNames: ['filter_practice_name_criteria','filter_method_name_criteria']};

// Initialize filter
var filterLinkPractices = new List('filter_link_practices', filterLinkPracticesOptions);

$('#filter_methods').change(
  function() {

    filterLinkPractices.filter(function(item) {

      if ($("#filter_methods").val() == "*") {
        filterLinkPractices.filter();
        return false;
      } else {
        if (item.values().filter_method_name_criteria.indexOf($("#filter_methods").val()) != -1) {
          return true;
        } else {
          return false;
        }
      }
    });
    return false;
  });

//Select/Unselect all checkboxes
$('#selectall').click(function(event) {
  if (this.checked) {
    // Iterate each checkbox
    $(':checkbox').each(function() {
      this.checked = true;
    });
  }
  if (!this.checked) {
    // Iterate each checkbox
    $(':checkbox').each(function() {
      this.checked = false;
    });
  }
});

/* ****************************************
 * Clear filters before submitting the form *
 * ****************************************/

// Get the form to link/unlink practices
```

```
var form = document.getElementById('form_link_practices_to_method');
// Save reference to original submit function
var formSubmit = form.submit;

form.onsubmit = function(e) {
  // Clear filters
  filterLinkPractices.search();
  filterLinkPractices.filter();
  // Submit form
  formSubmit();

  return false;
};
```

11 Fichiers de configuration

11.1 pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.unamur</groupId>
 <artifactId>AgileReferential</artifactId>
 <packaging>war</packaging>
 <version>0.0.1-SNAPSHOT</version>
 <name>AgileReferential Maven Webapp</name>
 <url>http://maven.apache.org</url>

 <!-- Properties -->

 <properties>
  <spring.version>4.0.3.RELEASE</spring.version>
  <spring.security.version>3.2.3.RELEASE</spring.security.version>
 </properties>

 <!-- Dependencies -->

 <dependencies>

  <!-- JUnit -->

  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
  </dependency>

  <!-- Spring -->

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-web</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-webmvc</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-tx</artifactId>
```

```
  <version>${spring.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>${spring.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-core</artifactId>
  <version>${spring.security.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>${spring.security.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>${spring.security.version}</version>
</dependency>

<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-taglibs</artifactId>
  <version>${spring.security.version}</version>
</dependency>

<!-- JSTL -->

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
</dependency>

<!-- Bean validation API -->

<dependency>
  <groupId>javax.validation</groupId>
```

```
    <artifactId>validation-api</artifactId>
    <version>1.1.0.Final</version>
   </dependency>

   <!-- Hibernate -->

   <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.3.4.Final</version>
   </dependency>

   <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>5.1.0.Final</version>
   </dependency>

   <!-- MySQL -->

   <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.30</version>
   </dependency>

   <!-- Log4j -->

   <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
   </dependency>

  </dependencies>

  <build>
   <finalName>AgileReferential</finalName>
   <plugins>
    <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-compiler-plugin</artifactId>
     <configuration>
      <source>1.6</source>
      <target>1.6</target>
     </configuration>
    </plugin>
   </plugins>
  </build>

</project>
```

11.2 app-agilia-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mvc="http://www.springframework.org/schema/mvc"
```

```
	xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:context="http://www.springframework.org/schema/context"
xmlns:tx="http://www.springframework.org/schema/tx"
	xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
  http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
beans.xsd
  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-
4.0.xsd
  http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">

	<!-- Enables Spring MVC @Controller annotation support -->
	<mvc:annotation-driven />

	<mvc:default-servlet-handler />

	<!-- Used to serve static resources like css and javascript files -->
	<mvc:resources mapping="/css/**" location="/media/css/" />
	<mvc:resources mapping="/js/**" location="/media/js/" />

	<!-- Scan classpath for annotations -->
	<context:annotation-config />
	<context:component-scan base-package="com.unamur" />

	<!-- Activates annotation based transaction management -->
	<tx:annotation-driven transaction-manager="transactionManager" />

	<!-- Resolves views selected for rendering by @Controllers to .jsp resources
		in the /WEB-INF/views directory -->
	<bean
		class="org.springframework.web.servlet.view.InternalResourceViewResolver">
		<property name="prefix" value="/WEB-INF/views/" />
		<property name="suffix" value=".jsp" />
	</bean>

	<!-- Hibernate Transaction Manager -->
	<bean id="transactionManager"
		class="org.springframework.orm.hibernate4.HibernateTransactionManager">
		<property name="sessionFactory" ref="sessionFactory" />
	</bean>

	<!-- Hibernate Session Factory -->
	<bean id="sessionFactory"
		class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
		<property name="dataSource" ref="dataSource" />
		<property name="annotatedClasses">
			<list>
				<value>com.unamur.entity.Method</value>
				<value>com.unamur.entity.MethodComment</value>
				<value>com.unamur.entity.Practice</value>
				<value>com.unamur.entity.PracticeComment</value>
				<value>com.unamur.entity.Role</value>
				<value>com.unamur.entity.User</value>
			</list>
		</property>
		<property name="hibernateProperties">
```

```
                        <props>
                                <prop key="hibernate.dialect">${hibernate.dialect}</prop>
                                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                                <prop
key="hibernate.connection.characterEncoding">${hibernate.connection.characterEncoding}</prop>
                        </props>
                </property>
        </bean>

        <!-- DataSource -->
        <bean id="dataSource"
                class="org.springframework.jdbc.datasource.DriverManagerDataSource">
                <property name="driverClassName" value="${jdbc.driverClassName}" />
                <property name="url" value="${jdbc.url}" />
                <property name="username" value="${jdbc.username}" />
                <property name="password" value="${jdbc.password}" />
        </bean>

        <!-- Properties files -->
        <bean
                class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
                <property name="location">
                        <value>/WEB-INF/config.properties</value>
                </property>
        </bean>

        <!-- MessageSource -->
        <bean id="messageSource"
                class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
                <property name="basename" value="classpath:messages" />
                <property name="defaultEncoding" value="UTF-8" />
                <property name="useCodeAsDefaultMessage" value="false" />
        </bean>

</beans>
```

11.3 app-agilia-security.xml

```
<beans:beans xmlns="http://www.springframework.org/schema/security"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/security
        http://www.springframework.org/schema/security/spring-security-3.2.xsd">

 <http auto-config="true" use-expressions="true">
  <form-login login-page="/login" default-target-url="/home?login=true" authentication-failure-
url="/login?error=true"
  username-parameter="email" password-parameter="password" />
  <logout logout-success-url="/login?logout=true" />
  <!-- URL with pattern /secure/** are only accessible by editors and administrators -->
  <intercept-url pattern="/secure/**" access="hasAnyRole('ROLE_EDITOR','ROLE_ADMINISTRATOR')"
  />
 </http>

 <authentication-manager>
  <authentication-provider>
```

```
    <!-- Set bcrypt as password encoder -->
    <password-encoder hash="bcrypt" />

    <!-- Select user and user role from database -->
    <jdbc-user-service data-source-ref="dataSource" users-by-username-query="select email, password, true
as enabled from user where email=?"
    authorities-by-username-query="select u.email, r.role FROM user u, role r WHERE u.role_id = r.id AND
u.email = ?"
    />
   </authentication-provider>
  </authentication-manager>

  <beans:bean
class="org.springframework.security.web.access.expression.DefaultWebSecurityExpressionHandler"
  />
</beans:beans>
```

11.4 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd">

   <display-name>Agilia</display-name>

   <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>

   <servlet>
    <servlet-name>app-agilia</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
    <servlet-name>app-agilia</servlet-name>
    <url-pattern>/</url-pattern>
   </servlet-mapping>

   <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
   </listener>

   <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/app-agilia-*.xml,
    </param-value>
   </context-param>

   <filter>
    <filter-name>SetCharacterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
     <param-name>encoding</param-name>
     <param-value>UTF-8</param-value>
```

```
    </init-param>
    <init-param>
      <param-name>forceEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>

  <filter-mapping>
    <filter-name>SetCharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

</web-app>
```

12 Fichiers de propriétés

12.1 config.properties

```
### JDBC Properties ###
jdbc.driverClassName= com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/AgileReferential?characterEncoding=UTF-8
jdbc.username=root
jdbc.password=admin

### Hibernate Properties ###
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.show_sql=true
hibernate.connection.characterEncoding=UTF-8
```

12.2 log4j.properties

```
### Root logger option ###
log4j.rootLogger=DEBUG, stdout, file

### Redirect log messages to console ###
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

### Redirect log messages to a log file, support file rolling. ###
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=C:\\log4j-application.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

12.3 messages.properties

```
# Texts for alert messages

alert.message.form.error = Veuillez corriger le(s) champ(s) indiqué(s) ci -dessous.
alert.message.login.error = Connexion refusée!Adresse e - mail et / ou mot de passe incorrect.
alert.message.login.success = Vous êtes à présent connecté.
alert.message.logout.sucess = Vous êtes à présent déconnecté.
alert.message.method.add.success = Une nouvelle fiche a été créée avec succès!
        Vous pouvez désormais l 'éditer.
alert.message.method.delete.success = La fiche a été supprimée avec succès!
alert.message.method.edit.success = La fiche a été modifiée avec succès!
alert.message.method.list.is.empty = Aucune méthode n 'est actuellement référencée dans le    système.
alert.message.practice.add.success = Une nouvelle fiche a été créée avec succès!
        Vous pouvez désormais l 'éditer.
alert.message.practice.delete.success = La fiche a été supprimée avec succès!
alert.message.practice.edit.success = La fiche a été modifiée avec succès!
alert.message.practice.list.is.empty = Aucune pratique n 'est actuellement référencée dans le    système.
alert.message.subscribe.success = Votre profil a été créé avec succès!

#Labels for buttons

button.add = Ajouter
button.cancel = Annuler
button.confirm = Confirmer
```

```
button.delete = Supprimer
button.edit = Editer
button.login = S 'identifier

# Breadcrumb labels

breadcrumb.home = Accueil
breadcrumb.login = S 'identifier
breadcrumb.method.add = Ajout d 'une méthode
breadcrumb.method.details = Détails d 'une méthode
breadcrumb.method.list = Liste des méthodes
breadcrumb.practice.add = Ajout d 'une pratique
breadcrumb.practice.details = Détails d 'une pratique
breadcrumb.practice.list = Liste des pratiques
breadcrumb.subscribe = S 'inscrire

# Simple labels

label.confirmation = Confirmation
label.description = Description
label.email = Adresse e - mail
label.firstname = Prénom
label.lastname = Nom
label.last.update = Dernière mise à jour:
label.name = Nom
label.password = Mot de passe
label.post.by = Par
label.post.on.date = Le
label.post.on.time = à

# Labels for modal titles

modal.title.method.edit.adoption.recommendations = Editer - Recommandations
modal.title.method.edit.bibliography = Editer - Références
modal.title.method.edit.description = Editer - Description
modal.title.method.edit.origins = Editer - Origines
modal.title.method.edit.scope = Editer - Champ d 'application
modal.title.method.link.practices = Associer / Dissocier des pratiques
modal.title.practice.edit.adoption.recommendations = Editer - Recommandations
modal.title.practice.edit.benefits = Editer - Bénéfices
modal.title.practice.edit.bibliography = Editer - Références
modal.title.practice.edit.common.mistakes = Editer - Erreurs courantes
modal.title.practice.edit.description = Editer - Description
modal.title.practice.edit.improvement.targets = Editer - Objectifs d 'amélioration
modal.title.practice.edit.observable.signs = Editer - Signes observables
modal.title.practice.edit.origins = Editer - Origines
modal.title.practice.edit.synonyms = Editer - Synonymes

# Labels for navigation

navigation.home = Accueil
navigation.login = S 'identifier
navigation.logout = Se déconnecter
navigation.methods = Les méthodes
navigation.practices = Les pratiques
navigation.subscribe = S 'inscrire
```

```
# Labels for panel titles

panel.title.adoption.recommendations = Recommandations
panel.title.benefits = Bénéfices
panel.title.bibliography = Références
panel.title.comments = Commentaires
panel.title.common.mistakes = Erreurs courantes
panel.title.description = Description
panel.title.improvement.targets = Objectifs d 'amélioration
panel.title.login = S 'identifier
panel.title.method.add = Ajout d 'une méthode
panel.title.method.list = Les méthodes
panel.title.observable.signs = Signes observables
panel.title.origins = Origines
panel.title.practices = Pratiques
panel.title.practice.add = Ajout d 'une nouvelle fiche
panel.title.practice.list = Les pratiques
panel.title.subscribe = S 'inscrire
panel.title.scope = Champ d 'application
panel.title.synonyms = Synonymes

# Labels for placeholders

placeholder.adoption.recommendations = Les recommandations d 'adoption de la pratique
placeholder.bibliography = Les références de la pratique
placeholder.comment = Ecrire un commentaire...
placeholder.method.description = Description de la méthode
placeholder.method.name = Nom de la méthode
placeholder.method.origins = Origines de la méthode
placeholder.method.scope = Champ d 'application de la méthode
placeholder.practice.benefits = Les bénéfices de la pratique
placeholder.practice.common.mistakes = Erreurs courantes de la pratique
placeholder.practice.description = Description de la pratique
placeholder.practice.improvement.targets = Les objectifs d 'amélioration de la pratique
placeholder.practice.name = Nom de la pratique
placeholder.practice.observable.signs = Les signes observables de la pratique
placeholder.practice.origins = Les origines de la pratique
placeholder.practice.synonyms = Les synonymes de la pratique
placeholder.search = Rechercher
placeholder.user.email = Adresse e - mail
placeholder.user.firstname = Prénom
placeholder.user.lastname = Nom
placeholder.user.password = Mot de passe

# Texts and labels for dropdown lists

select.filter.practices.check.all = Cocher toute la sélection
select.filter.practices.opt.all = Toutes les pratiques
select.filter.practices.optgroup = Filtrer les pratiques
select.filter.practices.prefix = Pratiques de

text.comments.none = Aucun commentaire disponible actuellement
text.home.baseline = Le référentiel des méthodes et pratiques agiles adapté à tous vos appareils.
text.home.welcome = Bienvenue sur Agilia!
text.info.none = Aucune information référencée.

#Texts for modal screens
```

```
text.modal.confirm.delete.method = Etes - vous sûr de vouloir supprimer cette fiche ?
text.modal.confirm.delete.practice = Etes - vous sûr de vouloir supprimer cette fiche ?

#Labels for page titles

title.home = Agilia - Accueil
title.login = Agilia - S 'identifier
title.method.add = Agilia - Ajout d 'une méthode
title.method.details = Agilia - Détails d 'une méthode
title.method.list = Agilia - Liste des méthodes
title.practice.add = Agilia - Ajout d 'une pratique
title.practice.details = Agilia - Détails d 'une pratique
title.practice.list = Agilia - Liste des pratiques
title.subscribe = Agilia - S 'inscrire

# Texts for tooltips

tooltip.method.bibliography = Où se renseigner pour en savoir plus ?
        (livres, articles, sites web)
tooltip.method.description = De quoi s 'agit-il ?
tooltip.method.scope = Quel est le champ d 'application ? Quelles sont les limites du champ d'
        application ?
tooltip.method.origins = Qui sont les auteurs ? Quel est le contexte historique ?
        Quelles sont les origines de la méthode ?
tooltip.method.practices = Quel groupe de pratiques constitue cette méthode ?
tooltip.method.recommendations = Quelques conseils à suivre lors de l 'adoption de cette méthode.
tooltip.practice.adoption.recommendations = Quelques conseils à suivre lors de l 'adoption de cette
        pratique.
tooltip.practice.benefits = Quels bénéfices confèrent au projet l 'utilisation de cette pratique ?
tooltip.practice.bibliography = Où se renseigner pour en savoir plus ?
        (livres, articles, sites web)
tooltip.practice.common.mistakes = Quelles sont les erreurs classiques commises ?
        Quels sont les mauvais usages courants ?
tooltip.practice.description = De quoi s 'agit-il ?
tooltip.practice.improvement.targets = Quels sont les objectifs d 'amélioration visés ?
tooltip.practice.observable.signs = Quels signes observables dans l 'espace de travail permettent de
        constater l'utilisation de cette pratique ?
tooltip.practice.origins = Qui sont les auteurs ? Quel est le contexte historique ?
        Quelles sont les origines de la pratique ?
tooltip.practice.synonyms = Quels synonymes connus désignent la même pratique ?

#Texts for errors

email.already.exists = Cette adresse e - mail est déjà enregistrée.
method.already.exists = Une fiche existe déjà pour cette méthode.
practice.already.exists = Une fiche existe déjà pour cette pratique.
NotEmpty.method.name = Ce champ est obligatoire.
NotEmpty.practice.name = Ce champ est obligatoire.
NotEmpty.user.firstName = Quel est votre prénom ?
NotEmpty.user.lastName = Quel est votre nom ?
Pattern.user.email = Veuillez entrer une adresse e - mail valide.
Size.user.email = Ce champ ne peut comporter plus de 128 charactères.
Size.user.firstName = Ce champ ne peut comporter plus de 64 charactères.
Size.user.lastName = Ce champ ne peut comporter plus de 64 charactères.
Size.user.password = Veuillez entrer un mot de passe comprenant au moins 6 caractères.
```

ANNEXE B

Interfaces graphiques du prototype réalisé

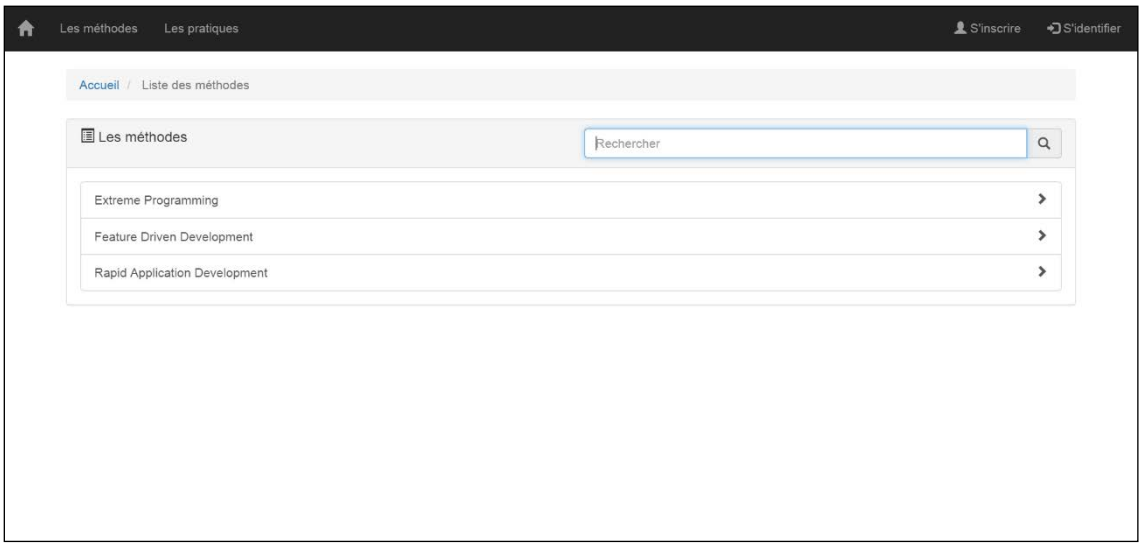
Table des matières

1 Tablettes et ordinateurs

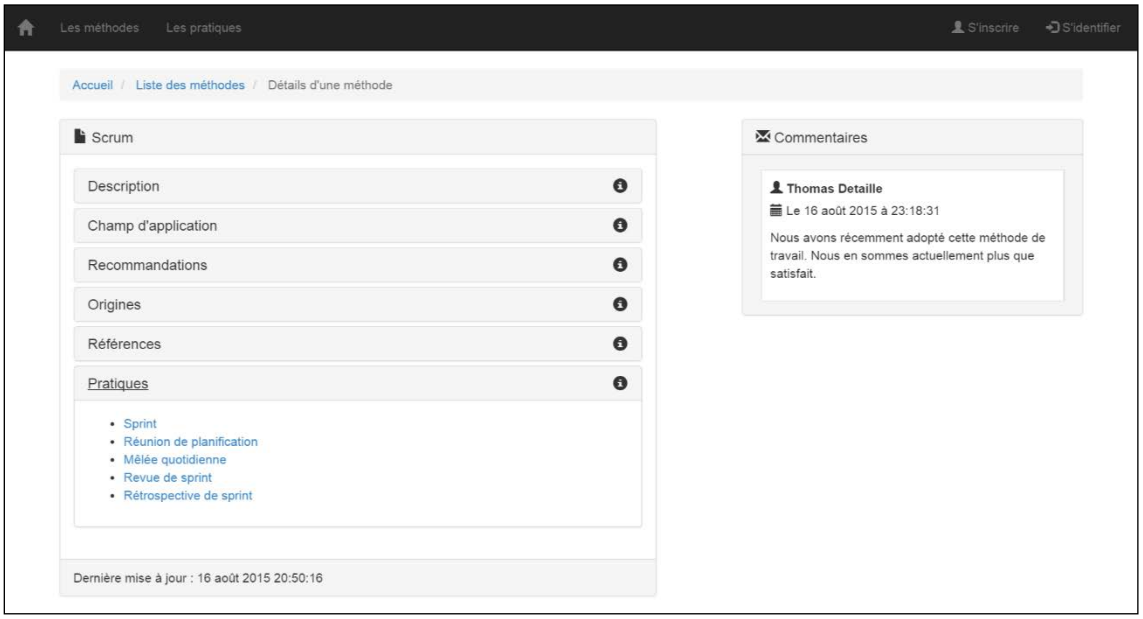
1.1 Mode anonyme

1.1.1 Accueil

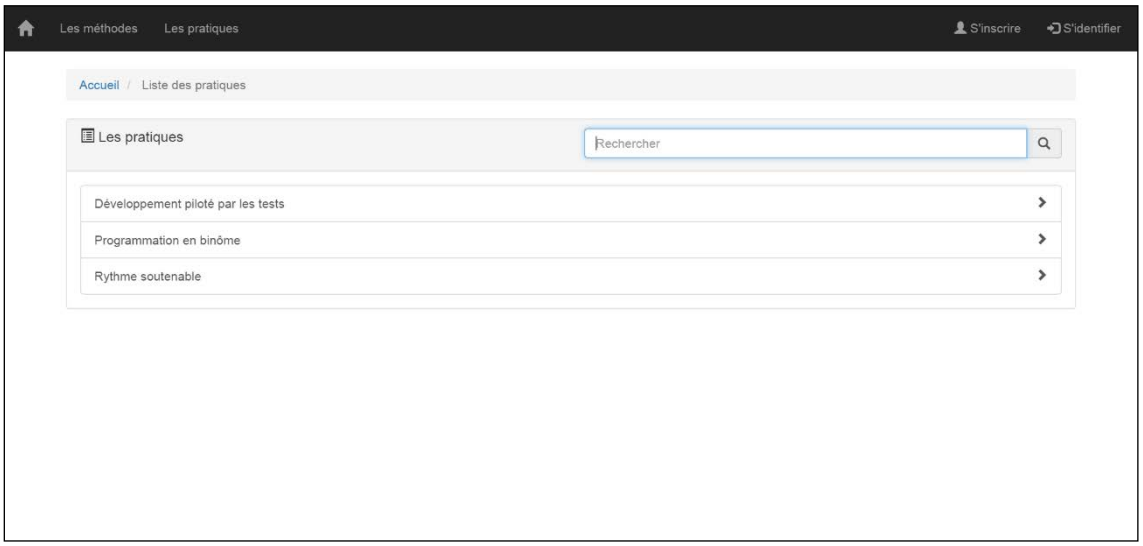
1.1.2 Aperçu des fiches de type méthode



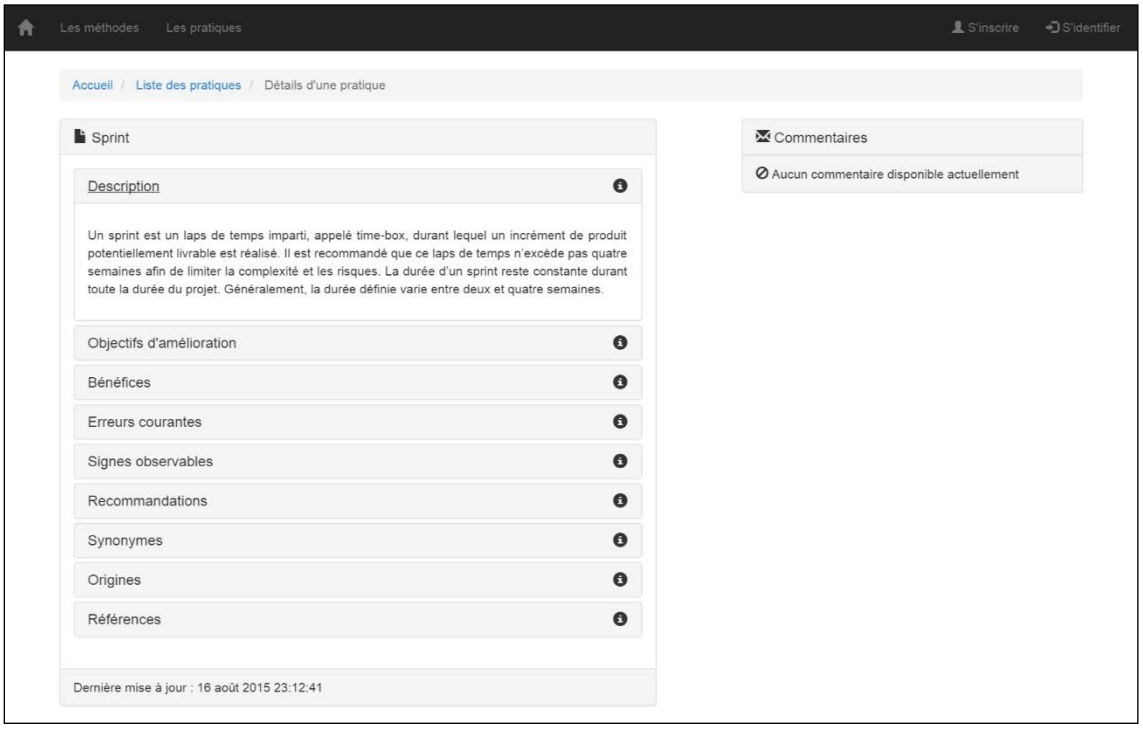
1.1.3 Détails d’une fiche de type méthode



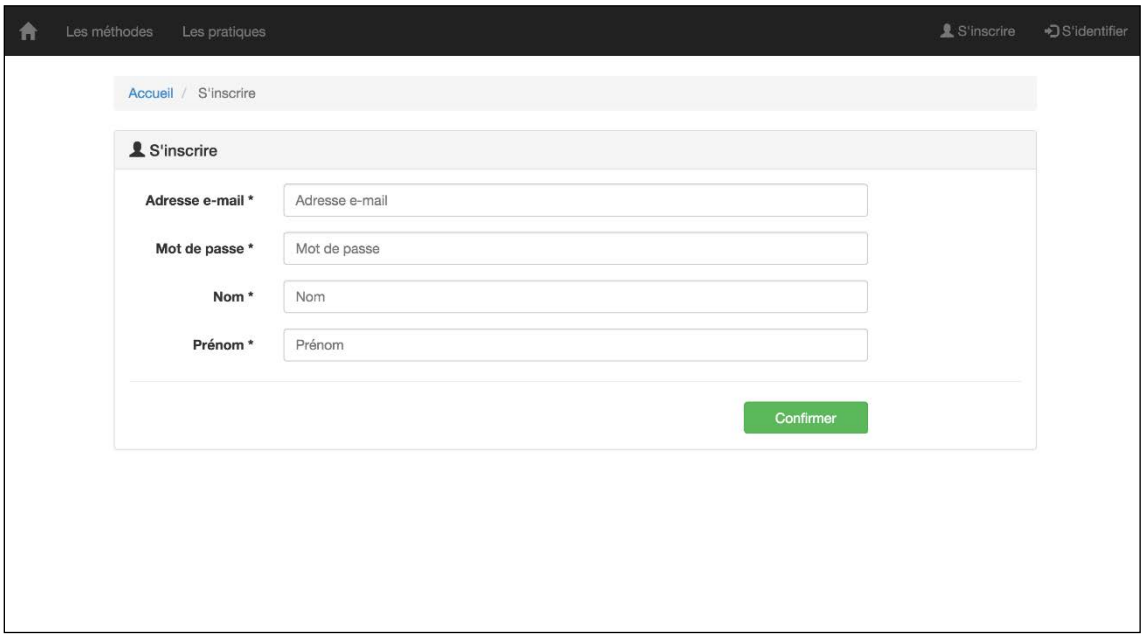
1.1.4 Aperçu des fiches de type pratique

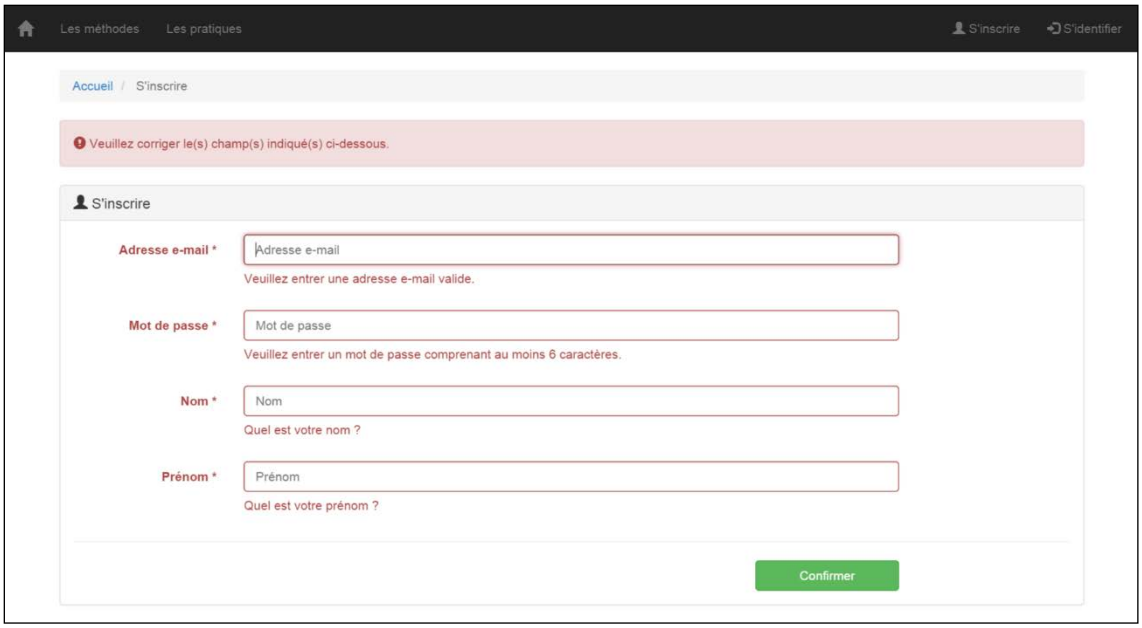


1.1.5 Détails d'une fiche de type pratique

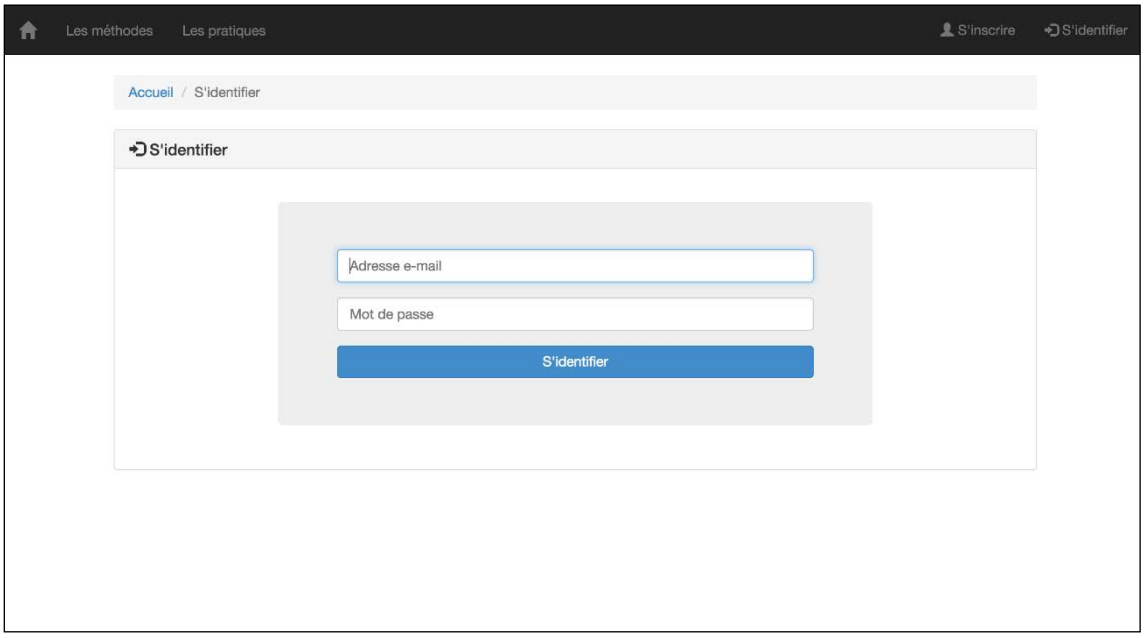


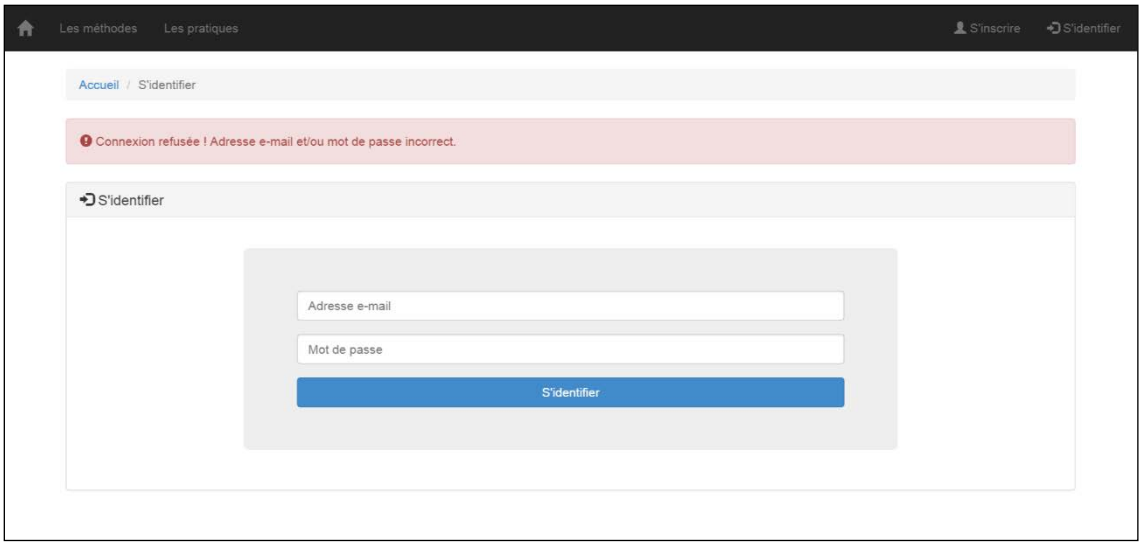
1.1.6 Inscription





1.1.7 Identification

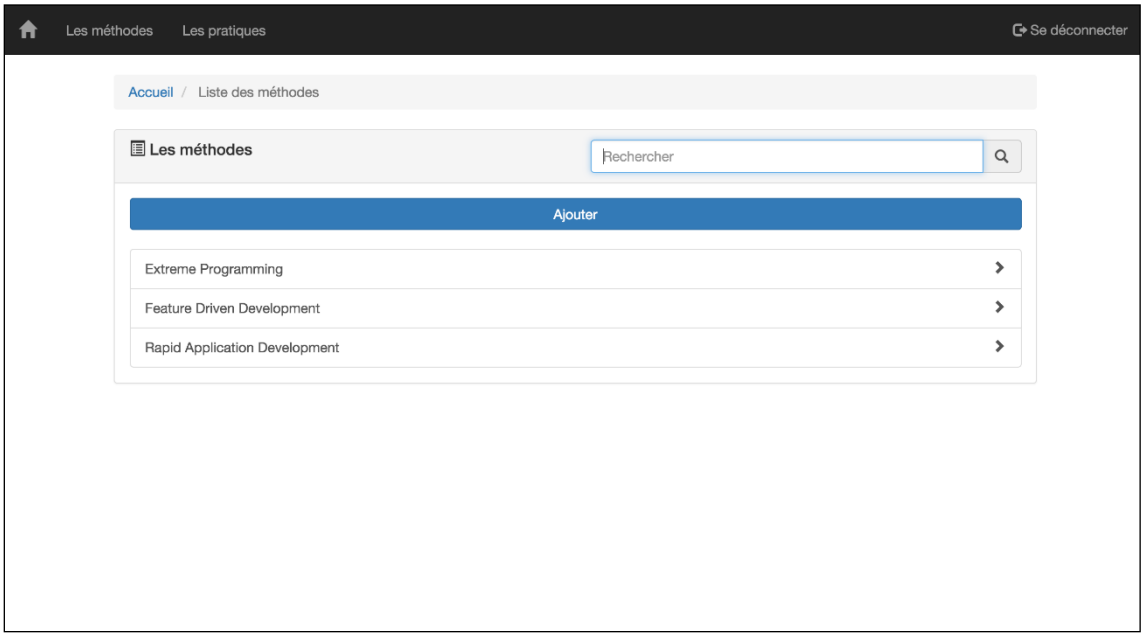




1.2 Mode authentifié

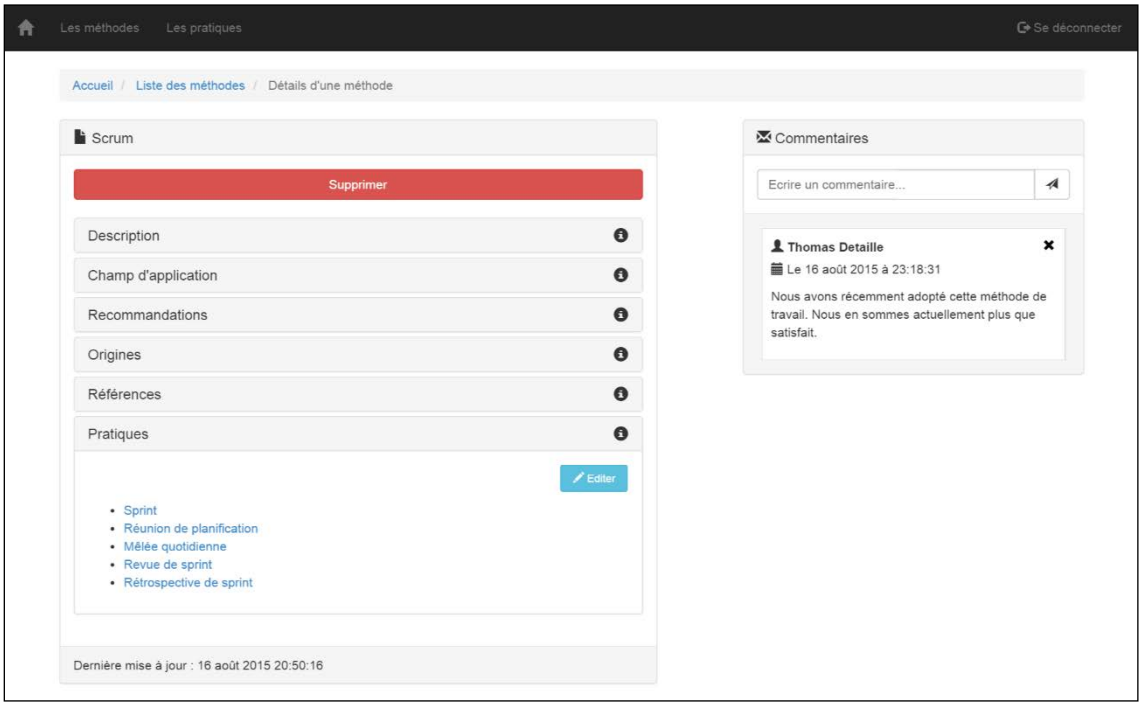
1.2.1 Accueil

1.2.2 Aperçu des fiches de type méthode



Note : Le bouton « Ajouter » est disponible uniquement pour les utilisateurs disposants droits d'éditeur ou d'administrateur.

1.2.3 Détails d'une fiche de type méthode

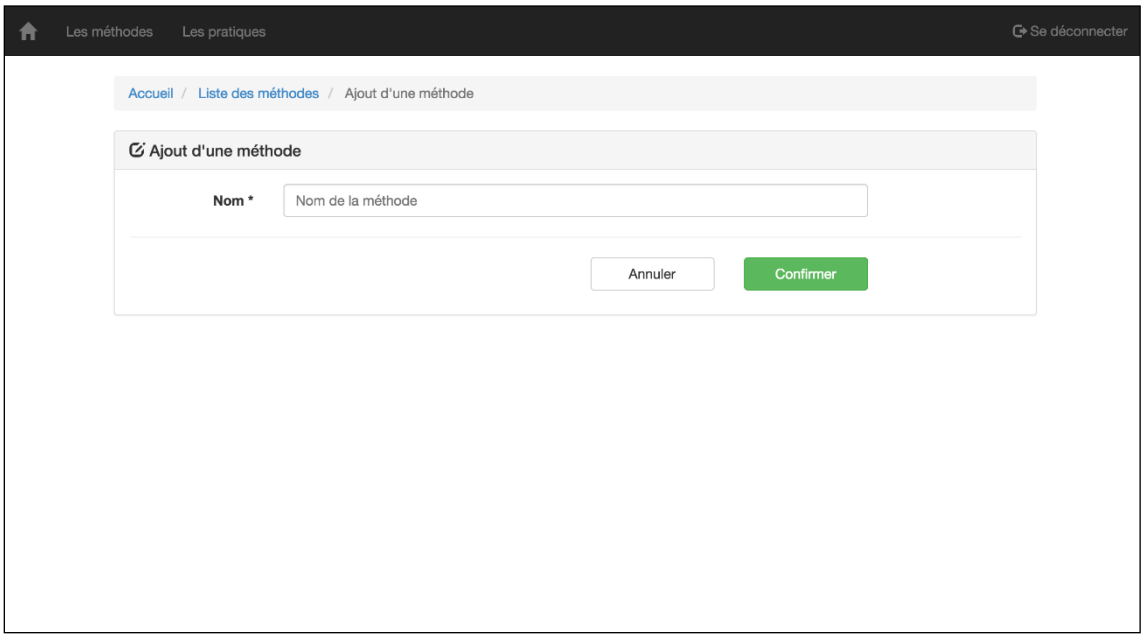


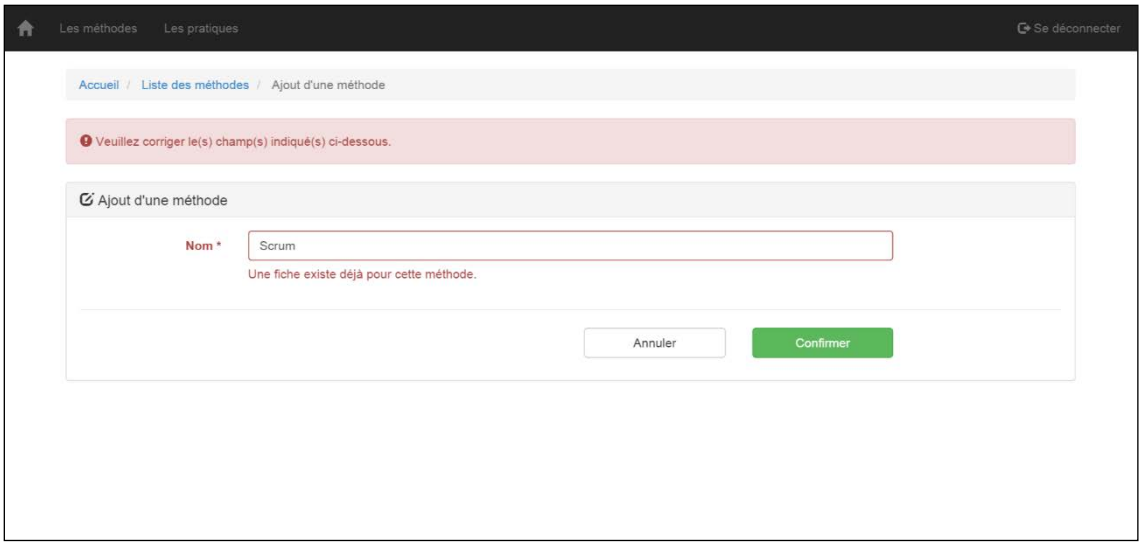
Note : Les boutons « Supprimer » et « Editer » sont disponibles uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

1.2.4 Edition d’une section d’une fiche de type méthode

Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

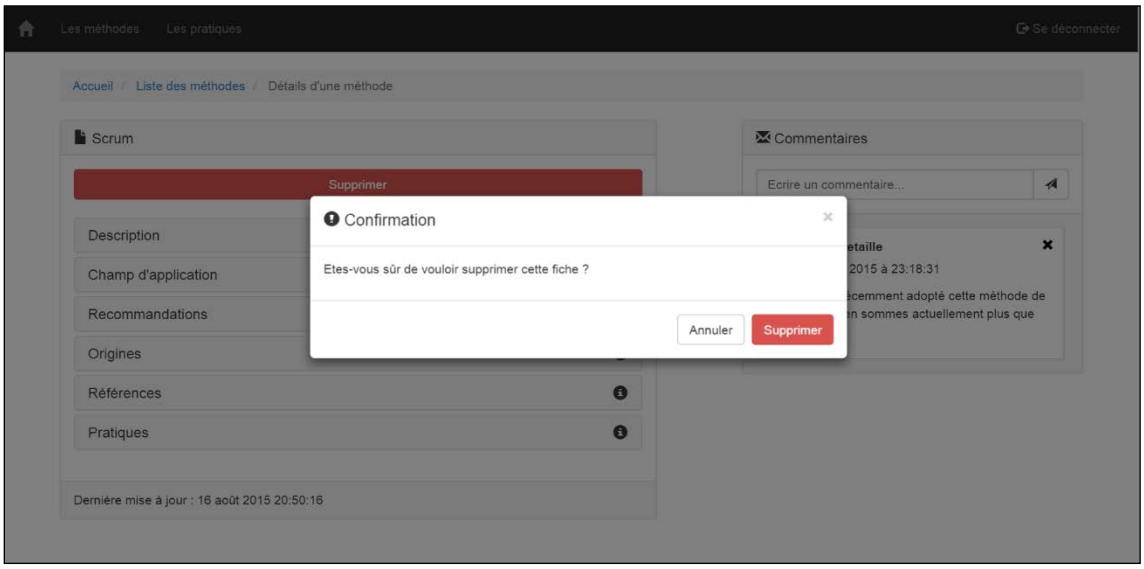
1.2.5 Ajout d’une fiche de type méthode





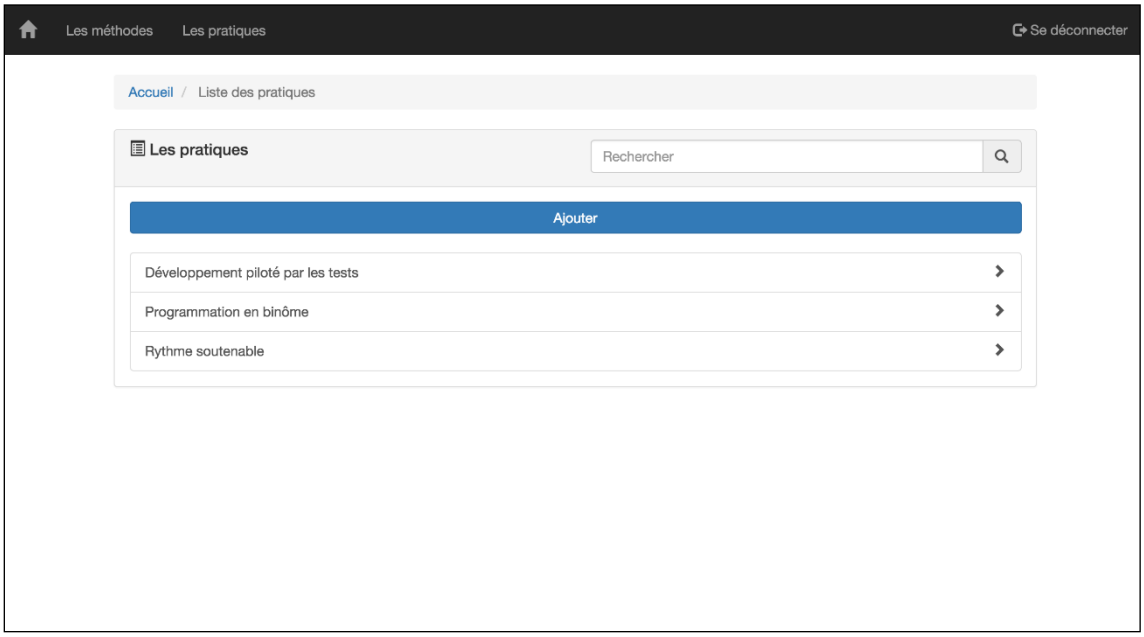
Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

1.2.6 Suppression d’une fichede type méthode



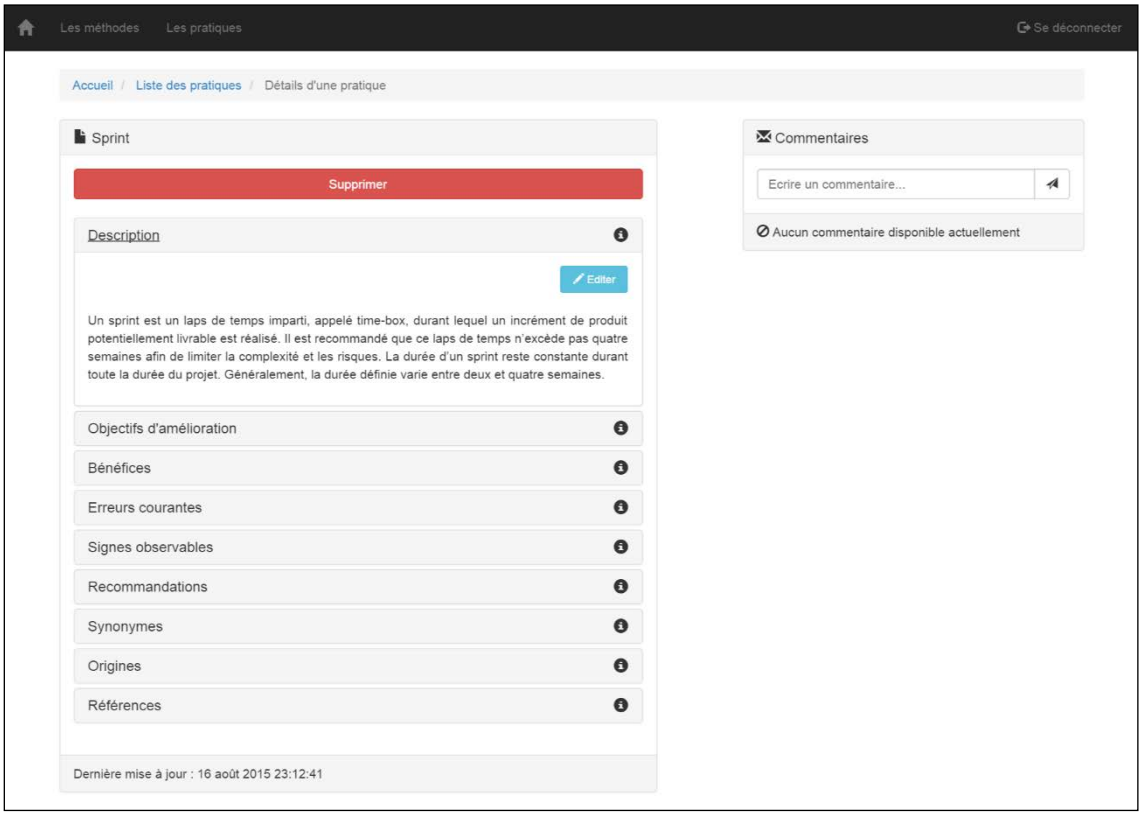
Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

1.2.7 Aperçu des fiches de type pratique



Note : Le bouton « Ajouter » est disponible uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

1.2.8 Détails d'une fiche de type pratique

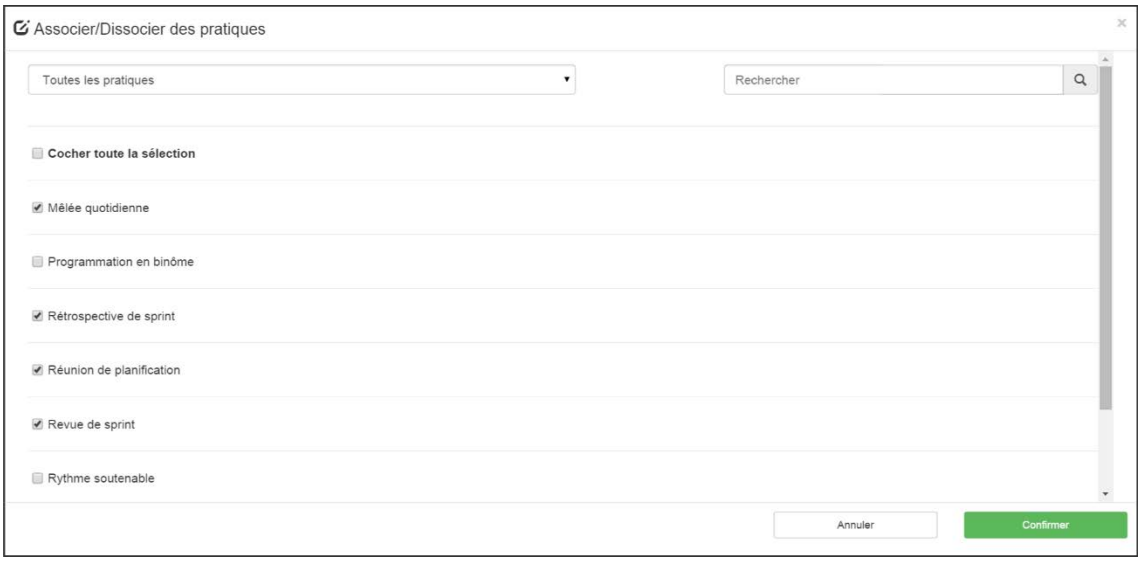


Note : Les boutons « Supprimer » et « Editer » sont disponibles uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

1.2.9 Edition d'une section d'une fiche de type pratique

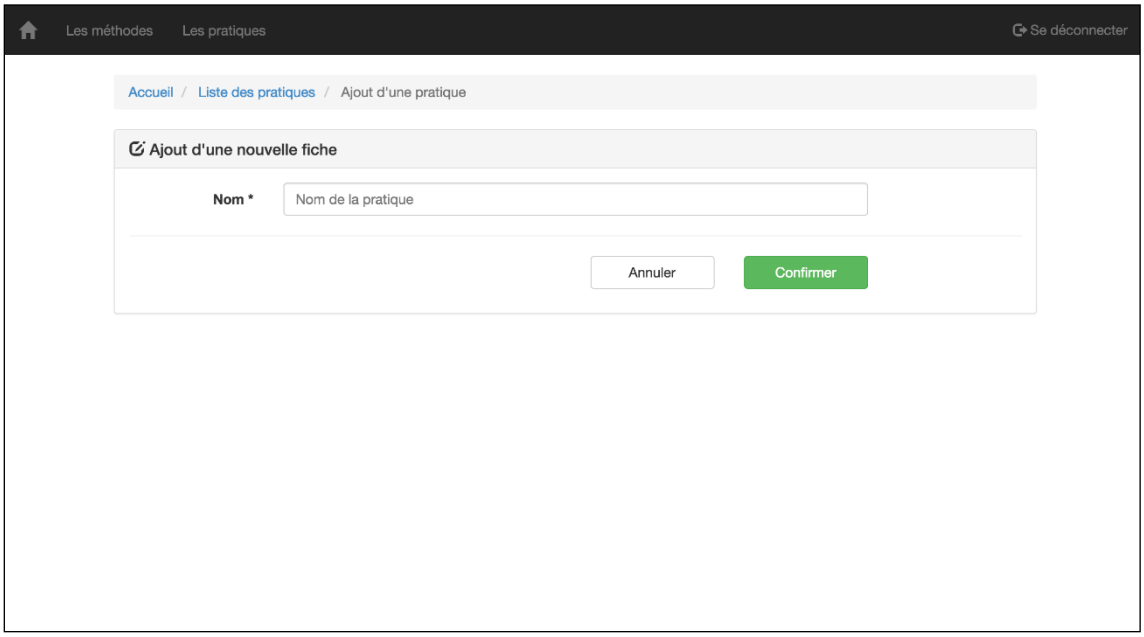
Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

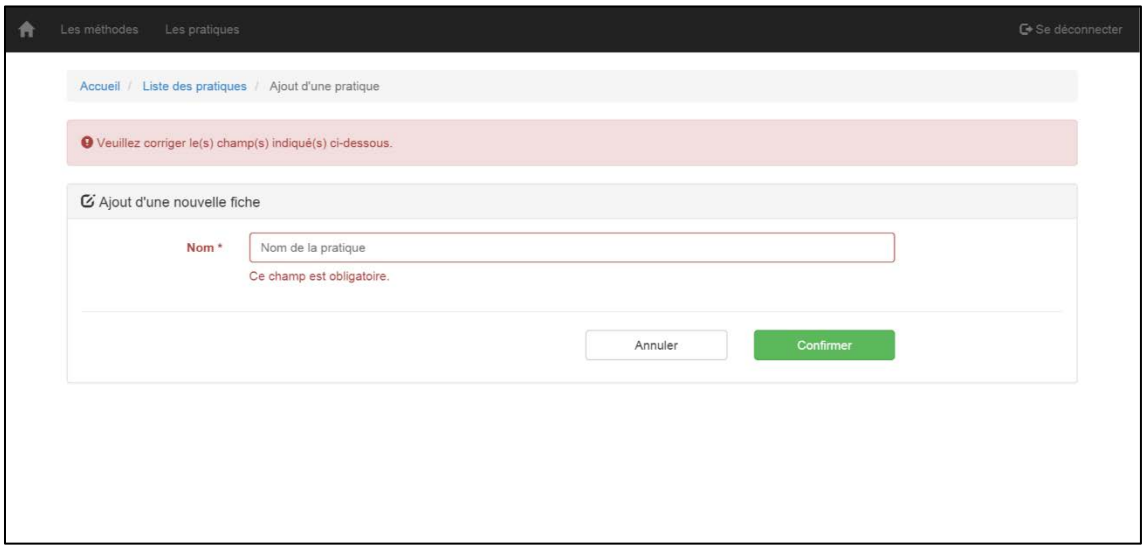
1.2.10 Association de fiches de type pratique à une fiche de type méthode



Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

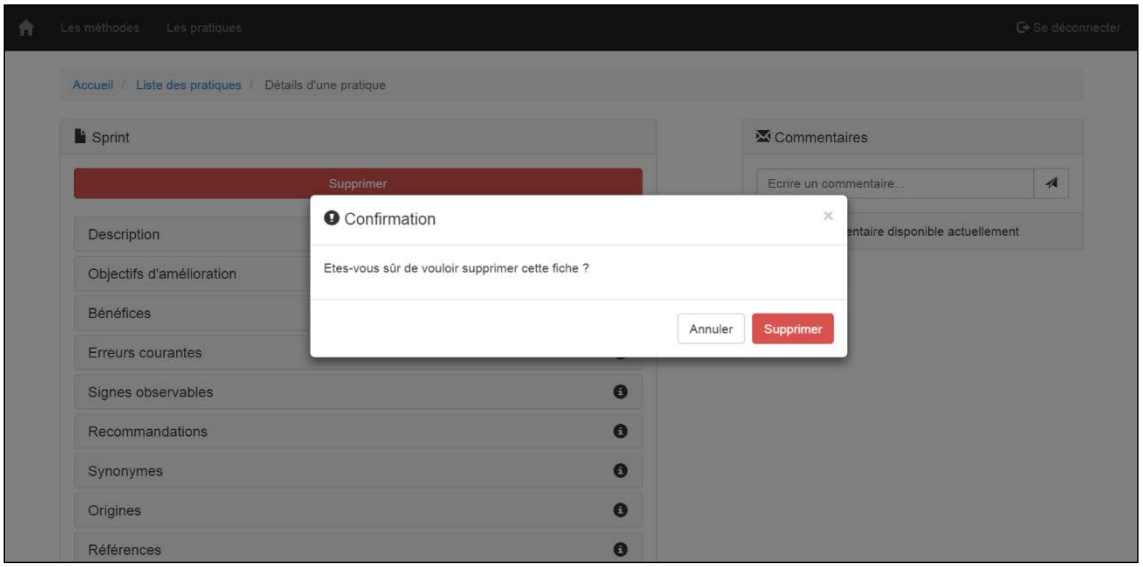
1.2.11 Ajout d’une fiche de type pratique





Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

1.2.12 Suppression d’une fiche de type pratique

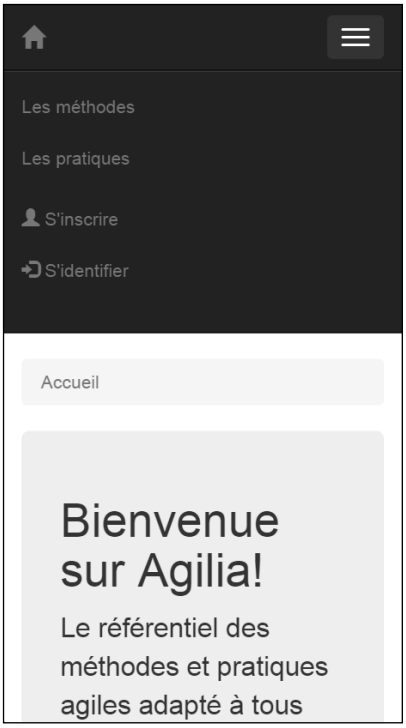


Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

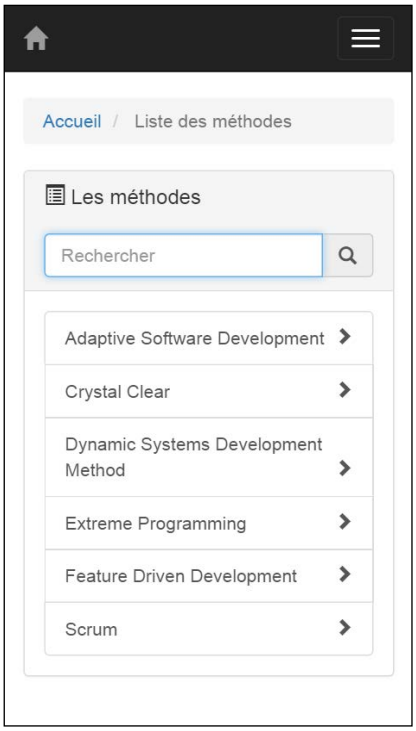
2 Smartphones

2.1 Mode anonyme

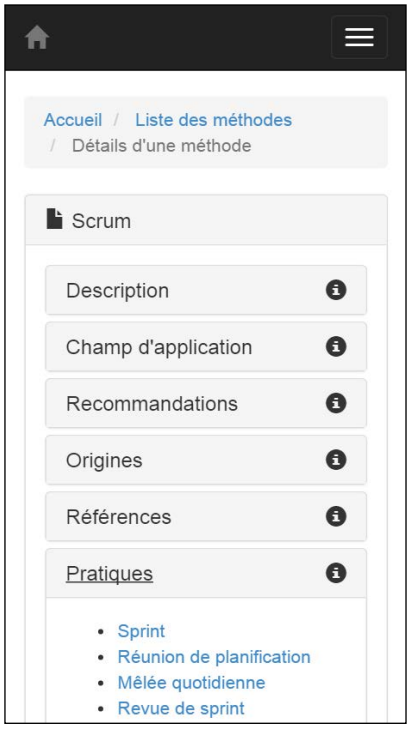
2.1.1 Accueil

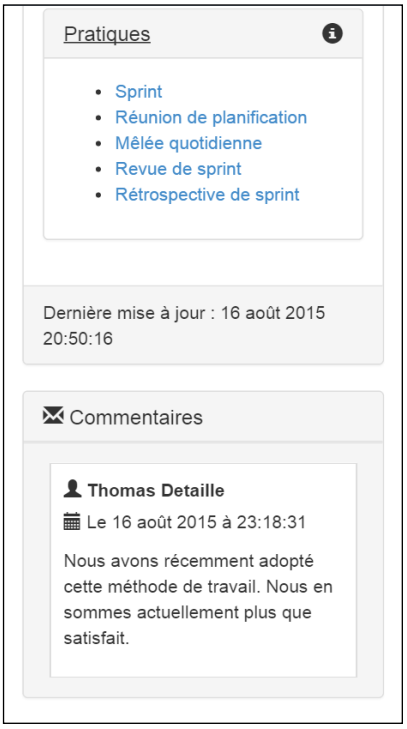


2.1.2 Aperçu des fiches de type méthode

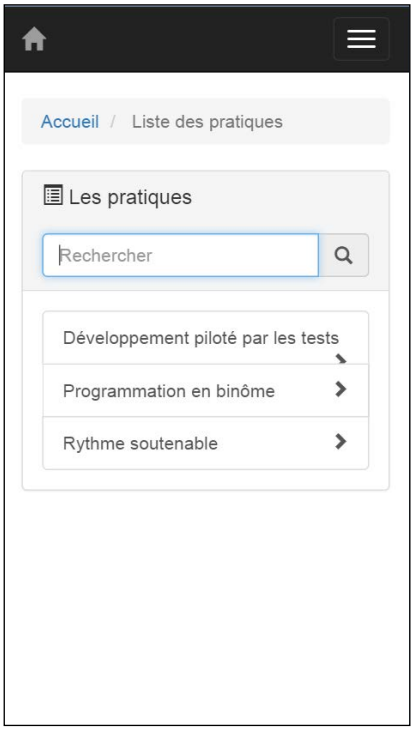


2.1.3 Détails d'une fiche de type méthode

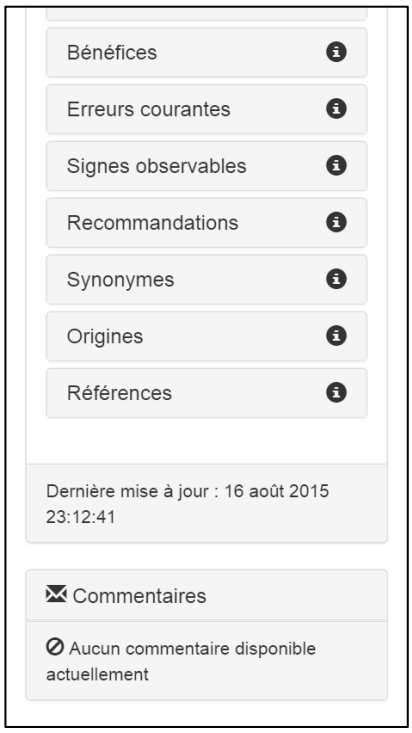




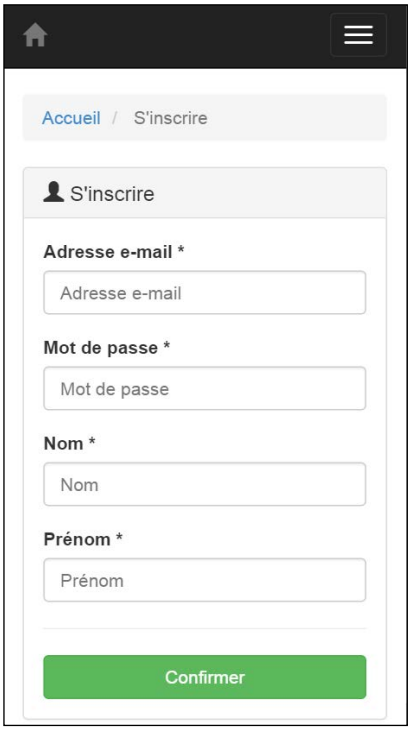
2.1.4 Aperçu des fiches de type pratique



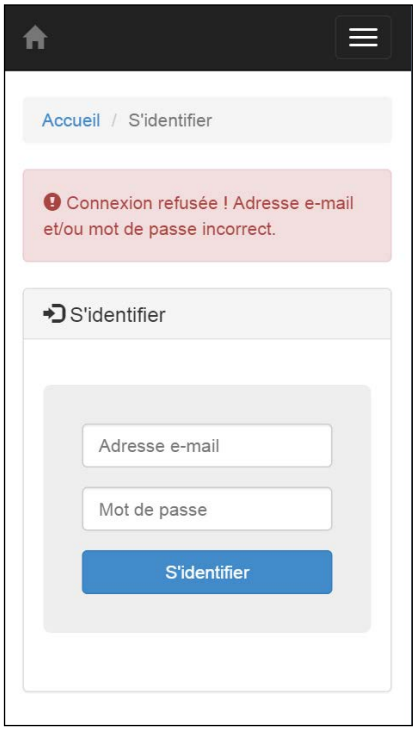
2.1.5 Détails d'une fiche de type pratique



2.1.6 Inscription



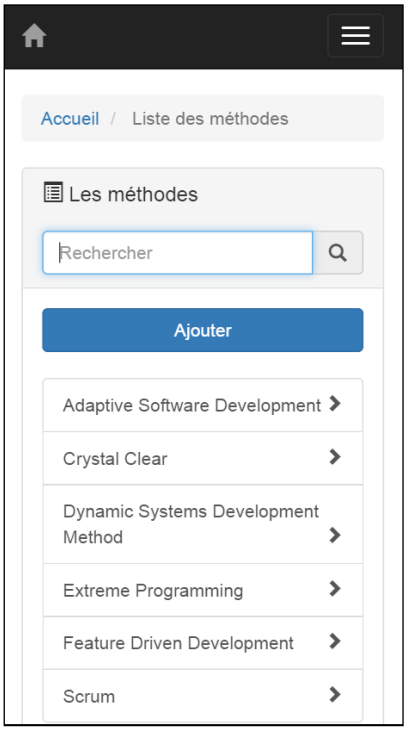
2.1.7 Identification



2.2 Mode authentifié

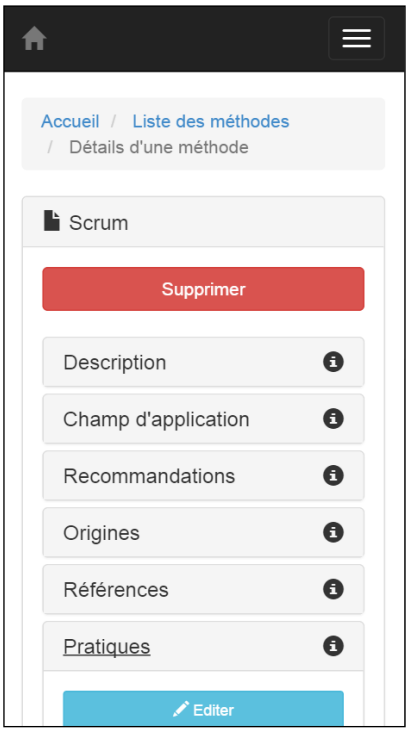
2.2.1 Accueil

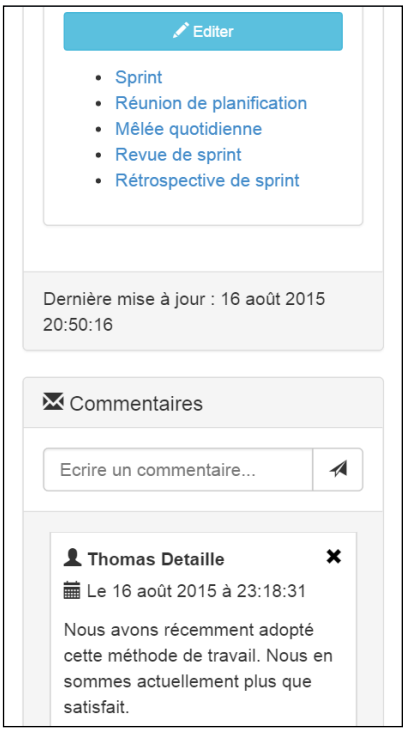
2.2.2 Aperçu des fiches de type méthode



Note : Le bouton « Ajouter » est disponible uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

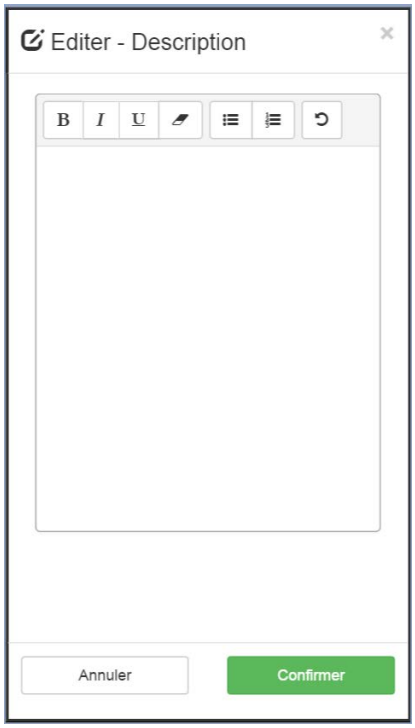
2.2.3 Détails d'une fiche de type méthode





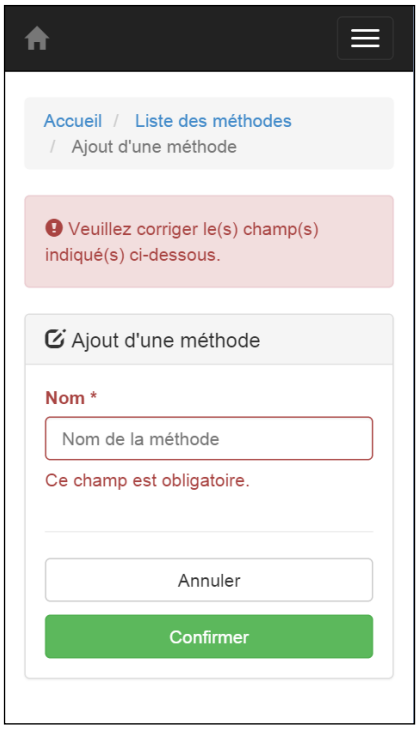
Note : Les boutons « Supprimer » et « Editer » sont disponibles uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

2.2.4 Edition d’une section d’une fiche de type méthode



Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

2.2.5 Ajout d’une fiche de type méthode

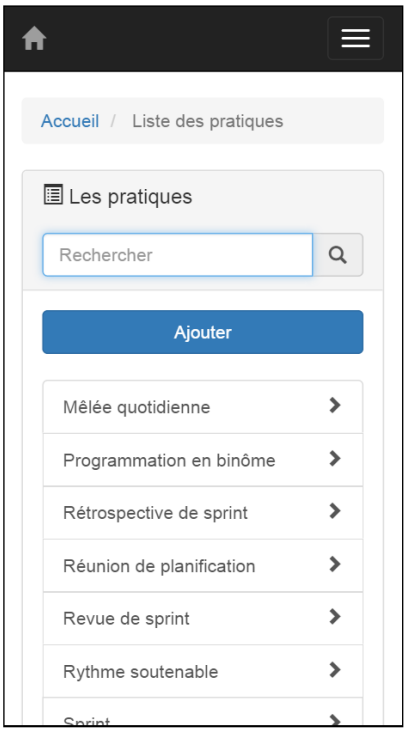


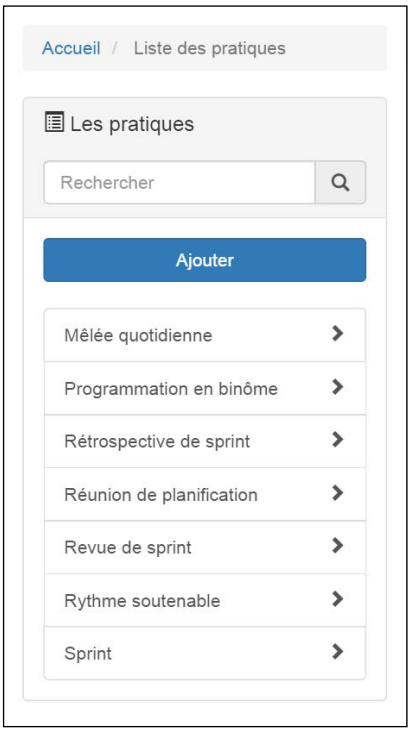
Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

2.2.6 Suppression d’une fiche de type méthode

Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

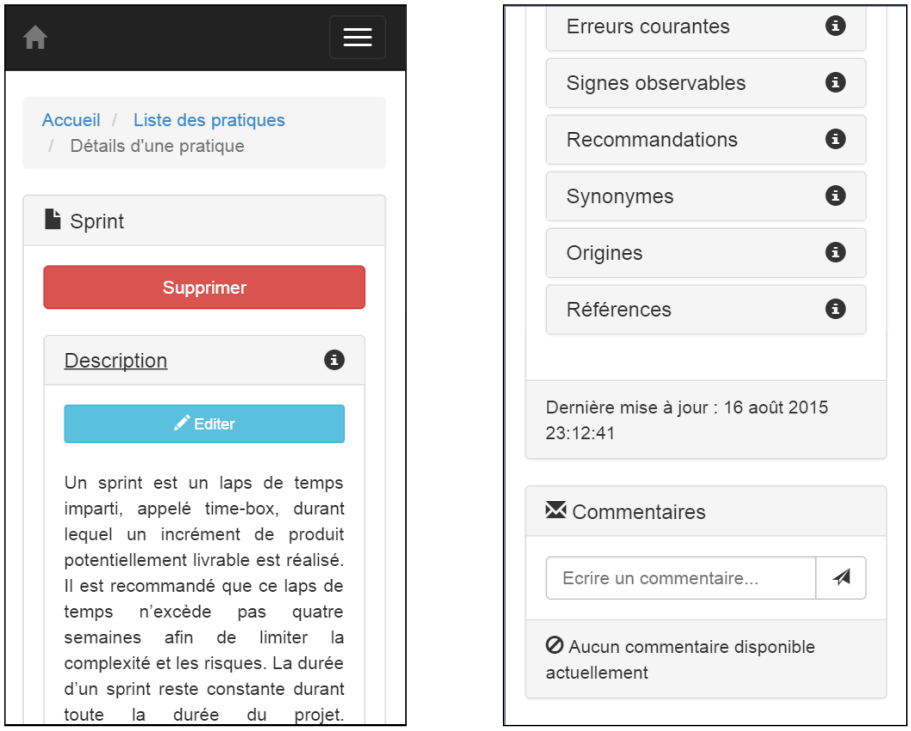
2.2.7 Aperçu des fiches de type pratique





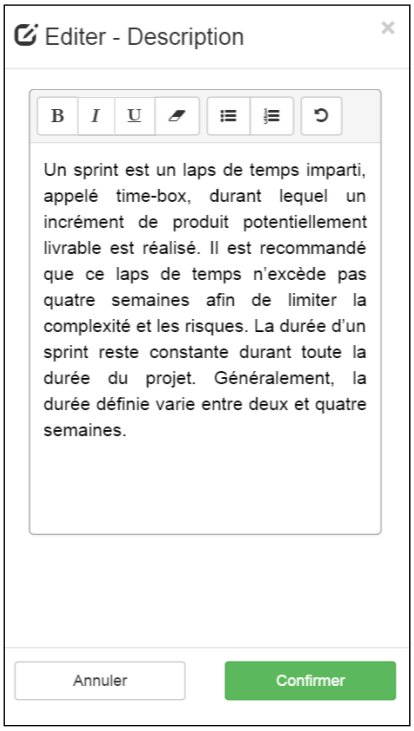
Note : Le bouton « Ajouter » est disponible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.

2.2.8 Détails d'une fiche de type pratique



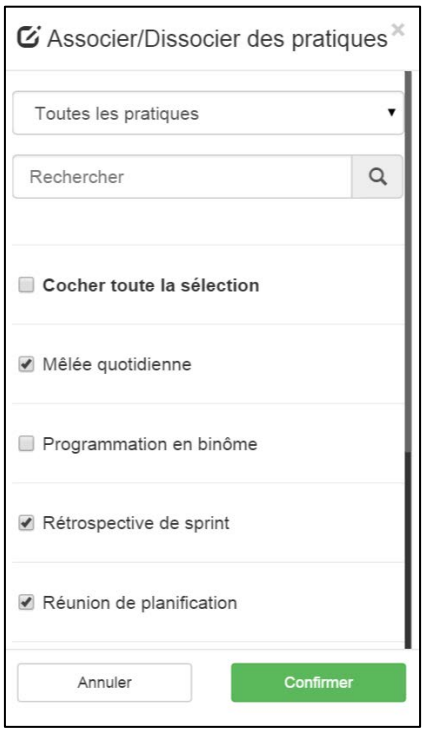
Note : Les boutons « Supprimer » et « Editer » sont disponibles uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

2.2.9 Edition d'une section d'une fiche de type pratique

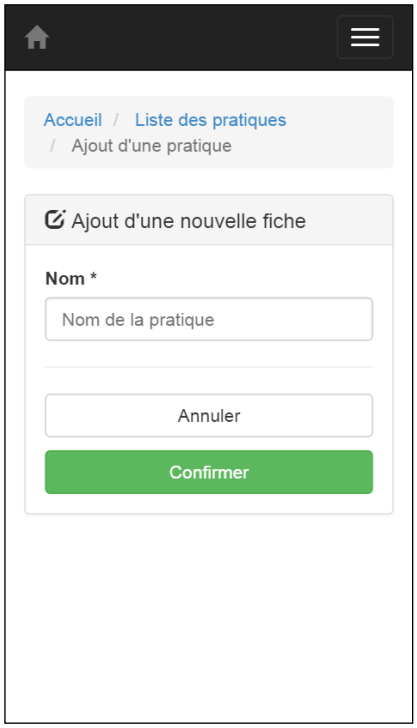


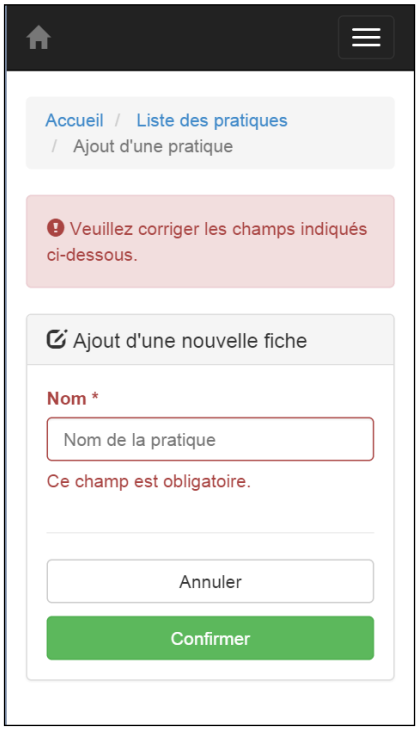
Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

2.2.10 Association de fiches de type pratique à une fiche de type méthode



2.2.11 Ajout d'une fiche de type pratique





Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d'éditeur ou d'administrateur.

2.2.12 Suppression d’une fiche de type pratique

Note : Cet écran est accessible uniquement pour les utilisateurs disposants des droits d’éditeur ou d’administrateur.